

HIERARCHICAL ATTRIBUTED GRAPH REPRESENTATION
AND RECOGNITION OF HANDWRITTEN CHINESE CHARACTERS

CENTRE FOR NEWFOUNDLAND STUDIES

**TOTAL OF 10 PAGES ONLY
MAY BE XEROXED**

(Without Author's Permission)

YING REN, B.Sc.



HIERARCHICAL ATTRIBUTED GRAPH REPRESENTATION AND RECOGNITION OF HANDWRITTEN CHINESE CHARACTERS

By

© Ying Ren, B.Sc.

A thesis submitted to the School of Graduate
Studies in partial fulfillment of the
requirements for the degree of
Master of Science

Department of Computer Science
Memorial University of Newfoundland
August, 1991

St. John's

Newfoundland

Canada



National Library
of Canada

Acquisitions and
Bibliographic Services Branch

385 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

385, rue Wellington
Ottawa (Ontario)
K1A 0N4

Author: *L' auteur*

Title: *Titre*

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-78132-7

Canada

ABSTRACT

This thesis presents a system which is capable of recognizing handwritten Chinese characters. The hierarchical attributed graph representation (HAGR), a two-level graph, is introduced to describe the structural and statistical information of handwritten Chinese characters. The first level describes radicals and relations between radicals within a character, the second level describes strokes and relations between strokes in a radical. With HAGR, the recognition process becomes a simple task of graph matching. A cost function mapping a candidate to a model graph is introduced. This approach can tolerate the variations of HAGR which reflect the instabilities and variabilities of handwritten Chinese characters resulting from different writing styles. Several rules have been used to re-arrange the order of the vertices of the graphs in order to avoid the combinatorial explosion inherent in graph matching. Based on HAGR, the model database is organized as a heterogeneous multi-way tree structure. For an input character, the search process can be divided into a number of simple and local decisions at different levels of the tree to find a corresponding model character in the database. The matching process is very efficient and accurate, and as well the system can acquire representations of characters by a learning process. Several HAGRs of samples of a character can be synthesized into a single HAGR of the character which can then be included in the model database. In addition, the learning process can update the models of characters with the HAGRs of their samples. The system is implemented in C on a MIPS/M-120 running RISC/OS (Version-3.1).

ACKNOWLEDGEMENTS

I wish to express my thanks to my supervisor Dr. Siwei Lu for his constant encouragement, insightful guidance, and constructive suggestion. Without his generous contribution, it would be impossible to give this thesis its current quality. I would like to thank the Systems Support staff for providing all the help and assistance during the conduct of my research. I am also very grateful to the Administrative staff who have helped in one way or another in the preparation of this thesis. In addition, I would like to acknowledge the financial support received from the Department of Computer Science and the School of Graduate Studies. Special thanks are due to my fellow graduate students and good friends, and in particular to Todd Wareham, Anthony Wing Key Szeto, Helmut Roth, and Sean Hogan for their valuable comments and useful suggestions. I would like use the chance to thank Prof. Jane Foltz, Patricia Murphy, and He Xu for their help and assistance.

*This thesis is dedicated to my parents and my sister
for their encouragement throughout
the course of my education.*

Table of Contents

Chapter 1	Introduction	1
1.1	Structure of the system	5
1.2	Organization of the thesis	5
Chapter 2	Survey of Techniques for Chinese Character Recognition	8
2.1	Introduction	8
2.2	On-line handwritten CCR	10
2.2.1	Feature analysis	10
2.2.2	Time sequence of zones, directions, or extremes	11
2.2.3	Curve matching	12
2.2.4	Stroke codes	13
2.2.5	Analysis-by-synthesis	14
2.2.6	Other methods	14
2.3	Handwritten and printed CCR	15
2.3.1	Statistical techniques	16
2.3.1.1	Template matching	16
2.3.1.2	Transformation	17

2.3.1.3	Stroke distribution	18
2.3.1.4	Background feature distribution	20
2.3.1.5	Background analysis	20
2.3.1.6	Stroke analysis	21
2.3.1.7	Combination schemes	22
2.3.2	Structural techniques	25
2.3.2.1	Grammar method	26
2.3.2.2	Graph methods	27
Chapter 3	Preprocessing	29
3.1	Introduction	29
3.2	Thinning	30
3.3	Tracing	35
3.4	Merging	39
3.5	Stroke grouping	45
3.5.1	Connection clustering	47
3.5.2	Rectangle clustering	47
3.5.3	Binary division clustering	48
3.5.4	Distance measurement clustering	49
Chapter 4	Attributed Graph Representation of Handwritten Chinese Characters	51

4.1	Introduction	51
4.2	Attribute sets and attributed graphs	53
4.3	The data structure of an attributed graph representation	59
4.4	Construction of hierarchical attributed graph	64
4.4.1	Radical attributed graph construction	64
4.4.2	Hierarchical attributed graph construction	65
Chapter 5	Recognition of Handwritten Chinese Characters	67
5.1	Introduction	67
5.2	Radical attributed graph matching	68
5.3	Example: finding a radical graph match	78
5.4	Ordering the vertices in a radical graph	81
5.5	Hierarchical attributed graph matching	85
Chapter 6	Model Database Organized by A Heterogeneous Multi-way Tree	89
6.1	Introduction	89
6.2	Heterogeneous multi-way tree organization	90
6.2.1	Basic concepts for the heterogeneous multi-way tree	91
6.2.2	Model database organized by multi-way tree	93
6.3	Construction of multi-way tree	101

6.4	Searching the multi-way tree for character recognition	103
6.4.1	Searching algorithm	101
6.4.2	Time complexity analysis of searching algorithms	109
6.4.2.1	Criteria for rejection of an input character	109
6.4.2.2	Criteria for recognition of an input character	110
6.5	Memory reduction	112
6.5.1	Memory reduction with radical linked list	112
6.5.2	Memory reduction with compression vectors	114
Chapter 7	Learning by Samples for Model Development	118
7.1	Introduction	118
7.2	Attributed graph synthesis	119
7.2.1	Synthesis evaluation of two RAGs	120
7.2.2	Synthesis evaluation of two HAGs	127
7.2.3	Synthesis of two radical attributed graphs	129
7.2.4	Synthesis of two hierarchical attributed graphs	130
7.2.5	Graph synthesis by ordering	131
7.3	Update the model database with the samples	133
7.3.1	Table network organization of model database	134
7.3.1.1	The host for characters	135

7.3.1.2 The host for radicals	136
7.3.2 RAG integration and HAG integration	138
7.3.3 Modification of models	141
7.3.3.1 Transformation	141
7.3.3.2 Updating	143
Chapter 8 Conclusions	152
8.1 Summary of contributions	152
8.2 Experiment and analysis	154
8.3 Directions for future research	158
8.3.1 Radical grouping	158
8.3.2 Error tolerating matching	158
8.3.3 Learning from feedback	159
References	160

List of Figures

Figure 1.1	Structure of proposed system	6
Figure 3.1	Eight neighboring pixels of a pixel n_0	31
Figure 3.2	Example of over erosion by Zhang-Suen algorithm	33
Figure 3.3	Comparisons of the thinning algorithms	35
Figure 3.4	Codes and directions for Freeman's chain coding	36
Figure 3.5	Encoding examples	37
Figure 3.6	Classification of the line segments	39
Figure 3.7	False stroke eliminating	41
Figure 3.8	A pattern of character with superimposed skeleton which contains a false stroke	43
Figure 3.9	A pattern of character with superimposed skeleton which contains a real stroke	44
Figure 3.10	Different classes of radicals	46
Figure 3.11	Example of rectangle clustering	48
Figure 3.12	Radicals that can be segmented with C-CT and R-CT	48
Figure 3.13	Stroke grouping	50



Figure 4.1	Structure relations between two adjacent radicals	52
Figure 4.2	Structural combinations of radicals	53
Figure 4.3	Radical attributed graph	56
Figure 4.4	A candidate character and its HAGR	58
Figure 4.5	Bits of the entries in the adjacency matrix of radical 	61
Figure 4.6	Using entry of adjacency matrix to represent the variations of stroke in the model	63
Figure 4.7	Construction of HAGR of a handwritten Chinese character	66
Figure 5.1	Vertex matching	69
Figure 5.2	Edge matching	71
Figure 5.3	Adjacency matrices A and B	74
Figure 5.4	MMC of A and B after exchanging the order of v_1 and v_3	75
Figure 5.5	Radical  and its model	79
Figure 5.6	Example for ordering strokes a and b using Rule 1	82
Figure 5.7	Example for ordering strokes a and b with Rule 2	83
Figure 5.8	Example for ordering strokes a and b with Rule 3	83
Figure 5.9	Example for ordering strokes a and b with Rule 4	84
Figure 5.10	Example for ordering strokes in a radical	85

Figure 6.1	Tree representing the partition of universal set	95
Figure 6.2	Tree representing the partition of a part into sections	96
Figure 6.3	Tree representing the partition of a section into groups	97
Figure 6.4	Tree representing the partition of a group	98
Figure 6.5	Representing character $\frac{\text{17}}{\text{X}}$ with a path in the tree	99
Figure 6.6	Heterogeneous multi-way tree organization of model database	100
Figure 6.7	Heterogeneous multi-way tree of character models	108
Figure 6.8	Radical list of <i>RAMS</i>	113
Figure 6.9	Compression vector	116
Figure 7.1	SRAGs of radical samples 力 and 扌	126
Figure 7.2	Reduced skeleton graph R_r of SRAGs R_1 and R_2	127
Figure 7.3	Convergent radical graph G_n of radical graphs R_1 and R_2	130
Figure 7.4	Character table	135
Figure 7.5	Radical table	136
Figure 7.6	Table network organization of model database	137
Figure 7.7	Example of RAG integration	138
Figure 7.8	Example of HAG integration	140
Figure 7.9	Transformation of samples for an input character	142

Figure 7.10	IIAG integration structure of an input character	147
Figure 7.11	Table network organization of database	148
Figure 7.12	RAG integration and IIAG integration in the updating	149
Figure 7.13	Updating results after step 8	150
Figure 7.14	Final updating results for the input character	151
Figure 8.1	Recognized samples	155
Figure 8.2	Character samples being rejected or misclassified	157

Chapter 1

Introduction

Handwritten Chinese character recognition is well known to be a very difficult problem. Because of its pragmatic value and its particularity in large template space recognition, the topic has attracted many researchers and has become one of the most challenging research subjects in the field of pattern recognition. The major difficulties come from the following factors: (1) There are more than forty thousand Chinese characters. Over a tenth of them are commonly used in daily life [Wang 1988]. Such a large number of character categories makes the recognition very difficult and slow. (2) The structures of Chinese characters are very complicated. Many of them contain more than thirty strokes. Furthermore, many characters are very similar to each other and are very difficult to distinguish. (3) Writing habits vary from person to person so that written characters have a great variety of character shapes, styles, positions of radicals, and directions or lengths of the strokes.

In spite of these difficulties, many techniques have been developed for solving the problem. They basically can be classified into two major approaches, namely, the statistical method and the structural method. Typical statistical approaches are correlation matching [Yamashita, et al. 1982], background feature distribution [Natio, et al. 1981], background analysis [Akamatus and Komori 1981], stroke analysis [Morishita, et al. 1988], and orthogonal expansion [Arakawa 1983]. Correlation matching typically requires some forms of normalization. However, normalization has only a limited capability to compensate for the large variety of writing styles, such as radical positions, direction and length of strokes. The last four approaches depend on extracted features such as stroke length, stroke direction, stroke sequence, and relation between strokes. Usually these features are represented as a feature vector and the character recognition is performed by selecting the reference character with the minimum distance from the input character. However, these features tend to be unstable as handwriting differs from person to person. One method to overcome such difficulties is to shift the burden to the users by imposing constraints on their writing [Wakahara and Umeda 1983]. However, not all users will accept this and some may have difficulties observing the specified rules. Another way is to increase the number of models for each character category in the database to allow for the variation caused by the instability of the features. Unfortunately, this will increase the database of the Chinese characters enormously. For example, the handwritten Kanji database FTL8 contains 152,960 samples of only 881 different Kanji characters. Each character cat-

egory has 160 samples written by different people [Nagy 1988]. With such a large character set, the matching process will be very time-consuming.

Although Chinese characters have a very complicated structure, they are structured according to some rules which are independent of writing styles. This structure can be divided into three levels: the whole character level, the radical level, and the stroke level. The geometric characteristics of the strokes vary to some extent with different writers, but their spatial relations and geometric configurations are usually well maintained. These properties can be regarded as invariant features and make structural approaches more attractive for the recognition of handwritten Chinese characters. One line of research within structural approaches represents a character pattern as a string and a grammar (either context-free or restricted context sensitive grammar such as an indexed grammar or a programmed grammar), and a parser for that particular grammar is built to recognize the pattern [Zhang and Xia 1983; Tai and Liu 1980; Zhao 1990]. String grammars are not powerful and flexible enough to handle very complicated characters with several radicals. Thus higher dimensional grammars (tree, plex, or web) are required. However, higher dimensional grammars are much more complicated and lack practical value. Another approach is to use pattern matching instead of parsing [Chan and Cheung 1989, Lee and Kim, 1989]. A character pattern is represented as a relational graph in which the vertices represent the strokes while the arcs represent the relationship between strokes. The graph approaches give a very flexible representation of structural pattern of Chinese

characters.

Despite these advantages, current graph approaches do not fully use the structural properties and do not provide effective organization of the huge model database for fast and accurate searching. For this reason, a new method which is not sensitive to writing styles is proposed and it offers users a high degree of flexibility for effective recognition of handwritten Chinese characters. This method is derived from the stable features of the characters. However, in the recognition process, some of the unstable features such as the orientations of the strokes are also necessary. Hierarchical attributed graphs are developed to describe both invariant and unstable features, with adjacency matrices used to represent these attributed graphs. The bits of the entries in a matrix describe the attributed set associated with a vertex or an edge. Based on the bit-wise representation, a cost function is introduced to map the graph of an input character to that of its model. This approach can provide some tolerance to the variations of characters written in different styles. For graph matching, several rules are applied to re-arrange the order of the vertices of the graph in order to avoid the combinatorial explosion. Furthermore, the model database is organized as a heterogeneous multi-way tree according to the spatial relations between radicals, the number of strokes per radical, and the geometric configuration of strokes in each radical. Using the hierarchical attributed graph representation and the multi-way tree organization of the database, the efficiency of the matching process can be improved considerably.

1.1 Structure of the system

The flowchart in Figure 1.1 gives an overview of the proposed system which consists of two functional parts. The first part is from the left-top box, "Input samples of a character", to the box "Table network organization of model database". The figure illustrates the procedure for building and updating the model database. The rest of the flowchart shows the second part which performs the recognition task. The recognition procedure may be divided into three levels: low, intermediate, and high. The low-level essentially involves thinning, skeleton tracing, segment merging, and stroke grouping. At the intermediate-level, the hierarchical attributed graph representation of the input handwritten character is generated. The recognition in the high level involves multi-way tree searching, graph matching, and mapping cost computation.

1.2 Organization of the thesis

The thesis is organized into eight chapters. Chapter two introduces existing techniques for recognition of Chinese characters such as machine printed Chinese character recognition, off-line Chinese character recognition, and on-line Chinese character recognition. Chapter three describes how the local properties of an input character are obtained and organized for further image analysis and representation. Chapter four is devoted to the hierarchical attributed graph representation of handwritten

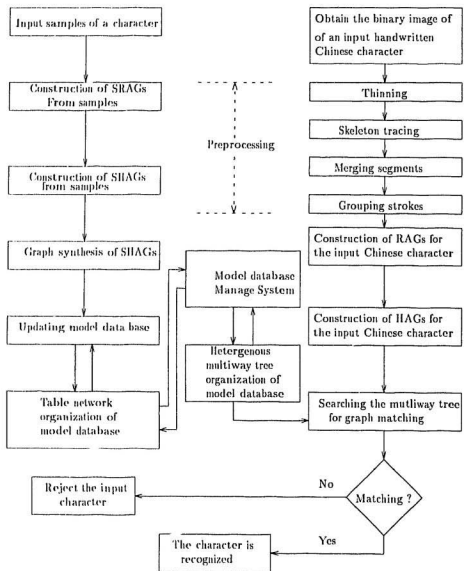


Figure 1.1 Structure of proposed system

Chinese characters and the construction of the hierarchical attributed graph representation. Chapter five gives the method for character recognition. A cost mapping function is introduced to match the attributed graph of an input character and that of its model. In Chapter six, the heterogeneous multi-way tree used to organize the model database for fast and accurate searching is discussed. Based on the multi-way tree, the search process can be divided into a number of simple and local decisions at different levels of the tree to find a corresponding character in the database. A learning procedure for building and updating the model database is described in Chapter seven. Chapter eight gives the conclusions and possible directions for further research.

Chapter 2

Survey of Techniques for Chinese Character Recognition

2.1 Introduction

Character recognition is a subset of pattern recognition. It was character recognition that gave the incentives for making pattern recognition and image analysis mature fields of science. Chinese character recognition (CCR) offered a challenge that was in certain key aspects representative of the larger world of pattern recognition. CCR provides an important way to input Chinese characters in a massive manner. In recent years, a considerable amount of work has been done on CCR. The CCR techniques can be classified into three main categories, namely, 1) printed CCR which is the recognition of specific Chinese fonts (Soong, Black, Kai, etc). 2) on-line handwritten

CCR which is the recognition of single handwritten Chinese characters, where not only the character image but also the timing information of each stroke is provided, 3) handwritten CCR which is the recognition of single handwritten Chinese characters which are unconnected and not written in calligraphy. So far, printed CCR and on-line handwritten CCR systems are already available in the market place. Techniques for doing multi-font printed CCR are available in the laboratories. Handwritten CCR, however, is still far from being practical [Leow 1987].

A typical CCR system includes the following functional components. During pre-processing, character patterns are digitized and converted into 2-dimensional binary images. Certain types of noise can be eliminated at this point. For on-line handwritten CCR, the stroke's position, direction, and length are captured while a stroke is drawn. Therefore, it is easier to obtain the stroke sequences of a Chinese character for the on-line CCR. Thinning is required to reduce the information needed for the next step. After preprocessing, the extracted features of the two-dimensional dot-matrix will be used to match against a set of pre-stored features of the referenced Chinese characters. The best matching will be used to identify the character. To reduce the recognition time and to achieve a high recognition rate, a multi stage recognition technology is applied in CCR. Preliminary classification is used to classify the large subset of Chinese character into small groups, then the final discrimination identifies the characters from each group.

In this Chapter, a brief overview of current approaches, techniques, and method

ologies in CCR is given. The problems of printed Chinese character recognition, on-line Chinese character recognition, and handwritten Chinese character recognition will be discussed.

2.2 On-line handwritten CCR

On-line handwritten CCR is simpler than printed CCR and off-line handwritten CCR because the machine can catch the accurate sequence of strokes. In on-line CCR, the system only needs to recognize less than one hundred different kinds of strokes. As long as the machine can recognize the strokes and the relationship of the strokes, it can recognize up to several thousands Chinese characters. Many methods are available for on-line classification of handwritten Chinese character. They are described below.

2.2.1 Feature analysis

A set of features can represent a handwritten Chinese character. The features might be based on the static properties of the character, the dynamic properties, or both. The features can be binary. With binary features, the name assigned to a known character is often determined by a decision tree [Hanaki, et al. 1976; Hanaki and Yamazaki 1980]. A disadvantage of this method is that it may not produce alternative character choices, which are usually desirable for postprocessing. Recently, a binary decision tree uses simple features to reduce the set of candidate characters to a small set for subsequent analysis by complex features [Kerrick and Bovik 1988]. The features

can also be nonbinary. A fixed number of nonbinary features is common in pattern recognition, and many classification methods are available for dividing such a feature space into decision regions. For example, a multi-stage classifier with general tree structure based on the dividing of Walsh coefficients has been developed [Gu, et al. 1983].

2.2.2 Time sequence of zones, directions, or extremes

These methods rely primarily on dynamic information. A sequence of coded zones can represent a character [Engdahl 1977; Iianaki and Yamazaki 1980]. The zones are specified by dividing up the rectangle that surrounds the written character, then the character is superimposed on the rectangle, and the sequence of zones traversed by the pen tip is determined. This sequence, or a corresponding sequence of features, assigns a name to the unknown character, often by exact match from a dictionary of zone sequences. A similar method uses the sequence of directions of pen tip motion during the writing of a character [Chang and Lo 1973; Crane and Savoie 1977; Groner 1968]. Using four primitive directions (up, down, left, right), one system coded the first four directions of the sequence and then classified the character by table lookup where the table had 256 entries [Groner 1968]. As the number of directions and time intervals increases, table lookup becomes less practical, and the sequences are compared by curve matching.

2.2.3 Curve matching

Curve matching is a popular image processing method. Curves from an unknown character are matched against those of prototype characters. The name of the prototype that best matches the unknown is assigned to the unknown. The curves are usually functions of time such as preprocessed x and y values, or the direction angle of the tangent to the trajectory of the writing [Ishigaki and Morishita 1988; Ishii 1986; Odaka, et al. 1986]. Using Freeman code, a character has been divided into ten regions [Li, et al. 1967]. Since Chinese characters consist mostly of straight strokes, approximating their strokes by a small number of fixed points has been found successful [Odaka, et al. 1982]. An alternative to the matching of functions of time is the matching of Fourier coefficients obtained from the $x(t)$ and $y(t)$ curves [Arakawa, et al. 1978; Impedovo 1984]. This method is appropriate when the characters can be represented by a reasonably small number of Fourier coefficients. Since straight-line strokes require high-order Fourier coefficients, this method has been useful for characters consisting mostly of curved strokes, like numerals, or of concatenations of many straight strokes, like Chinese characters [Arakawa, et al. 1978]. Curve matching becomes equivalent to pattern matching in feature space when the number of points characterizing the curve is constant and there is a one-to-one correspondence [Odaka, et al. 1982]. This is a linear alignment of the points of the curve. However, due to nonlinearities, the best fit is usually not a linear matching or alignment. For many sequence comparison problems, *elastic matching* has been successful [Ikeda, et

al. 1978; Sato and Adachi 1985; Wakahara and Umeda 1983; Yoshida and Sakoe 1982]. Because elastic matching is computationally intensive, the prototypes are required to be first pruned to reduce the computation. Application of a local affine transformation can enhance the shape discrimination of elastic matching. Using the point correspondence from elastic matching between input and reference patterns, a deformation vector field is generated and then approximated by means of iterative applications of local affine transformations. Finally, further elastic matching between the input pattern and the deformed reference pattern superimposed by low order local affine transformation components enhances shape discrimination, halving the error rate [Wakahara 1988].

2.2.4 Stroke codes

The stroke code method classifies subparts of a character and then identifies the character from the sequence of classified subparts [Greanias and Yhap 1982; Hing-Hua 1988; Kuo, et al. 1988; Lin and Tsai 1988]. One system uses 76 stroke codes of constituent shapes to specify and recognize more than three thousand Kanji characters [Yurugi, et al. 1985]. Stroke classification uses the sequence of direction angles. Then decision trees of stroke code sequences under relative positional constraints on strokes classify the radical or character. Another system uses a formalism based upon an initial stroke-sequence decision tree and position matching [Chen, et al. 1988]. This formalism has the advantage of using the features of strokes, stroke-sequence, and

geometric relations but avoids the disadvantages caused by the instability of all of the above features.

2.2.5 Analysis-by-synthesis

Yet another approach is analysis-by-synthesis, sometimes called recognition-by-generation. Several studies are concerned with the modeling of handwriting generation [Yasuhara 1975; Morishita, et al. 1988]. These methods usually use strokes and rules for connecting them to build characters. Characters generated from the inventory of strokes constitute idealized standard representations of the characters. An approximation to real handwritten characters can be attained by specifying these strokes with mathematical models that describe the motion of the pen tip as a function of time. Then a handwritten character can be divided into strokes. The strokes are classified using the model parameters, and the stroke sequences [Yoshida and Eden 1973]. A similar approach uses dynamic programming to match real and modeled strokes [Wakahra and Umeda 1984]. Due to its optimality property, dynamic programming can be used to obtain the minimum distance between an input and a reference pattern to handle the problem caused by the distortion existing in the input pattern.

2.2.6 Other methods

Perceptual studies have been instrumental in the development of pairwise distinction methods which separates each pair of characters that might be confused. Studying

the way humans distinguish between such pairs led to a theory of characters based on functional attributes [Cox, et al. 1982; Watanabe, et al. 1985]. Pair distinction by functional attributes has led to robust recognition methods, notably that in the commercial system by Percept. Sometimes the same attribute differentiates more than a pair of characters [Sakai, et al. 1984]. Another method represents a character by the number, order, and relative position of strokes; some strokes are divided into more parts, particularly those of characters with few strokes [Kato, et al. 1978]. The statistical method of Markov models is particularly suitable for dynamic information. [Farak 1979] uses a first order Markov model with eight states corresponding to eight pen-tip directions. A system unifies the statistical and syntactical approaches for on-line Chinese character recognition [Tai and Liu 1990]. A fuzzy attributed finite-state automaton is introduced for stroke recognition. According to the intrinsic structure of Chinese characters, a two-dimensional character is transformed into a one dimensional attributed string on the basis of order arrangements of Chinese characters. Such strings can be easily recognized by template matching.

2.3 Handwritten and printed CCR

Handwritten (off-line) and printed CCR are more difficult than on-line CCR because the former one is performed after the writing or printing is completed and therefore has no temporal or dynamic information such as number of strokes, order of the strokes, direction of the writing for each stroke, or speed of the writing within each

stroke. Moreover, handwritten CCR is the most difficult aspect of character recognition, because the noises at elements and the distortions in structures are dealt with simultaneously, especially the large scope of distortions produced by different writers. The techniques used in handwritten and printed CCR can be roughly divided into statistical and structural approaches. The statistical decision or decision-theoretic approach involves the use of transformation functions, distribution decision functions or their equivalent functions, such as Bayesian classifier, statistical equivalent block classifier, and so on. The syntactical or structural approach uses various two-dimensional grammars for character description, parsers for analyzing the structure of an unknown character, and attributed graphs for describing character components and components relationships.

2.3.1 Statistical techniques

Template matching

The earliest approach for CCR was reported twenty five years ago [Casey and Nagy, 1966]. The authors used one of the simplest pattern recognition techniques: template-matching. Each character is assigned a template or mask which is a matrix of black and white pixels. To classify a given character sample, its matrix is compared to all templates. Classification is achieved if one of the templates provides a sufficiently good match to the character sample. To speed the matching, a two stage matching process was introduced. That is, similar characters were grouped first, then masks

were grouped and finally individual masks were employed. In general, this method involved expensive pixel-by-pixel comparison of the matrix of the input character and the template. In addition, such method is only applicable to printed characters in which the size and position of the radicals can be almost constant and stable. In the case of handwritten characters, normalization of a character does not necessarily mean normalization of a radical which constitutes the subpattern of the character.

Transformation

Fourier, Hadamard, and KL (Karhunen-Loeve) transformations have been applied to printed Chinese character recognition [Nakata, et al. 1972; Gu, et al. 1983; Sakai, et al. 1976; Leung 1985], but only KL has been used for both handwritten and printed CCR. One of the most attractive properties of the two-dimensional Fourier transform is its ability to recognize position-shifted patterns since it observes the magnitude spectrum and ignores the phase. It is well recognized that the precision of center-location is a problem for the scanner, and it is anticipated that will also be a problem for identifying printed Chinese characters. The Hadamard transform is more acceptable in high-speed processing since its arithmetic computation involves only addition and subtraction. The major drawback of application of this technique in pattern recognition is that its performance depends too heavily upon the position of the pattern.

In particular, KL was very successful in printed CCR, in which three orthogonal axes were used in order to absorb the variations of displacement and width of lines.

However, more than ten such axes are needed for handwritten CCR. Furthermore, it is not practical due to the heavy computation involved in diagonalizing the $N^2 \times N^2$ correlation matrix corresponding to the sample images digitized on a $N \times N$ grid [Leung 1985].

For Chinese character patterns represented by their outer contour, Fourier descriptors are very useful in recognition [Krzyzak and Buaeshi 1989]. Among different techniques, Fourier descriptors are distinguished by their invariance relative to the standard shape transformations such as scaling, rotation, translation, and mirror reflections. The major drawbacks of Fourier descriptors are (1) their insensitivity to spurs on the boundary and (2) disconnected patterns give a completely different spectrum and style variations are reflected in the lower order spectrum [Verschueren, et al. 1984].

Stroke distribution

A distribution of local stroke features can be taken in a two-dimensional plane or projected on a one-dimensional axis. A popular example is to consider the line directions as the local stroke features. Such a scheme is called direction matching [Yasuda and Fujisawa 1979; Saito, et al. 1982]. The boundary direction is calculated at each boundary point by following the contour between two pre- and post-points along a binary pattern on a 64×64 grid plane. The direction is quantified into four directions and mapped to a 16×16 plane. This method is quite simple, however the recognition rate is very low. For improvement, size normalization and shift similarity should be

used. Another example of stroke feature distribution on the two-dimensional plane uses stroke length [Hagita and Masuda 1981] which is considered as a special case of the distance representation introduced in the field effect method [Mori, et al. 1974]. At each black point, eight quantized directions are taken and the distance is measured along each direction from this center point to the boundary point. Then two distance values along two opposite directions are summed and a four-dimensional distance vector can be obtained at each pixel on a 128×128 plane. To get a compact feature distribution, this plane is divided into 8×8 zones, each of which is of 16×16 , and the vectors are averaged over each zone. Matching is done by the so-called MDD rule (minimum distance decision) which is essentially the same as correlation. This scheme extracts more global features than a local directional feature so that some complex features such as intersection points are reflected. However it cannot distinguish the characters which are very similar to each other.

An almost identical method was used which tried to recognize printed Chinese character on the license plates of moving vehicles [Dai, et al. 1988]. A standard X and Y profile was defined for each character, and for a given character sample, its profiles were constructed and compared to all standard profiles. The criterion for recognition was the pair of profiles yielding the closest correspondence. This approach yields some advantages: 1) Using two one-dimensional patterns per character as opposed to one two-dimensional pattern results in considerable information reduction. 2) The projection profiles are easily extracted from the original pattern. 3)

Since the projection profiles are obtained by an integrative process, they tend to be less sensitive to noise. However, the projection profiles suffered from position errors between the input and standard patterns. In addition, different characters with the same projection files cannot be discriminated.

Background feature distribution

The background feature distribution technique [Suen 1982; Naito, et al. 1981] is based on a slight modification of Gluckman's well known method of background features extraction [Gluckman, 1967]. For every background picture element of a binary pattern, scanning lines are derived in four directions, top, bottom, left, and right. In each scanning, the number of crossings between the scanning line and strokes is counted. However, this does not give an exact stroke density in each direction, because four quantized directions are not necessarily perpendicular to the strokes; sometimes they cross the strokes tangentially or even in parallel. For improvement, a crossing is counted only when each direction is nearly perpendicular to a stroke.

Background analysis

Instead of propagating black and white information as in Gluckman's method, more exact information on strokes being propagated can be extracted. The idea is to propagate edges, namely edge value and direction [R. Oka 1982; Yamamoto 1984]. In this method, a cell is defined on cellular space with meshes of 7×7 and each cell has eight intracells for eight directions. Each intracell stores the strength of the

edge (edge value) whose direction is just normal to the direction of this intracell. These intracell features represent geometric features of the input pattern around the cell. For each cell, the edge values of all its intracells are averaged. Only those cells whose averaged edge value exceeds specified threshold value are selected and used in matching, which is carried out by distance measurement. However, as the image quality of the input character varies, it is difficult to select an unequal threshold value even for two different quality images of the same input character.

Stroke analysis

Stroke analysis is the most traditional approach for Chinese character recognition. Generally, stroke analysis is based on the skeleton obtained by the thinning preprocessing, but it is well known that such results are not satisfactory because of noise and distortion [Kimura, et al. 1978]. For printed Chinese characters, the typical types of noise are distorted intersection points, whiskers, and branches caused by touching strokes, since the strokes of printed characters are usually very thick. There still remains a major problem of stroke segmentation after thinning. However, thinning preprocessing is still attractive because of the simplicity of the algorithm. Basic research continues on thinning algorithms and their application to stroke segmentation as well [Pavlidis 1982; Wakayama 1982; Liao and Huang 1990].

On the other hand, to admit that not all of the noise sources mentioned can be removed, a practical approach is to consider some non-local but still simple noise removal method, which would be effective against the major types of noise. In this

sense, an idea of the so called "good continuity rule of Gestalt psychology" is very useful to remove such noise as distorted intersections. In fact, this rule is used in a very simple way of removing noise and detecting strokes [Kasvand 1979]. All pairs of segments joining at an intersection point are needed, with segment continuations being measured according to some conditions. The pair of segments with maximum continuity which is greater than some threshold value is chosen as one stroke and the rest are treated in the same manner.

For extraction of the stable strokes, a technique using Hough transform (HT) was proposed recently for handwritten CCR [Cheng, et al. 1989]. First the character pattern is thinned and transformed from the spatial domain to the parametric one by IIT. As most strokes of Chinese characters are almost linear, they can be easily detected as lines by IIT within the heavy noise image. This is a new approach to the application of HT and a new attempt at the stroke extraction of handwritten Chinese character. The method is still very time-consuming as no preclassification exists to reduce the number of matching characters by using those features obtained by HT.

Combination schemes

For the recognition methods based on feature extraction, each character category is made by finding the reference vector with the least distance from the input pattern. Motivated by the requirement of seeking more effective and more reliable features, much research effort has been made and various character features have been proposed. Nevertheless, it is apparent that none of these features can yield sufficient

accuracy when used alone. This problem is caused by two inherent drawbacks common to all of the features.

One of these drawbacks is the lack of discriminatory information. Theoretically speaking, the purpose of feature extraction is to assure reliability of recognition by removing redundant and irrelevant information and enhancing the separability among pattern classes. In practice, features are often extracted by means of some measurements of the character pattern. An overall effect of feature extraction is that much redundant information is removed, but for a small number of characters, some important discriminatory information is lost. In the Chinese character set, there are many pairs of similar characters that differ from each other only slightly. If such a significant difference is ignored by the measurement, ambiguity will arise between these characters. Worse is the fact that some dissimilar characters may have very similar feature vectors. If the reference vectors of the characters are crowded closely in the feature space, recognition would be very difficult. For example, the feature vector of input pattern may deviate from its reference at a small distance, as is often the case for a somewhat noisy character sample, and be closest to a reference vector of another character category, resulting in a mis-recognition that is difficult to avoid.

Another essential weakness of the character features lies in their low stability against noise or disturbance. In the character samples read from actual printed documents, there may exist many kinds of disturbance, such as blurred stroke, broken stroke, positional shift, character rotation, stroke thickness variation, and noisy

points. Every feature is particularly sensitive to disturbances which can considerably affect the results of the measurements on which the feature is based. Under such a disturbance, a great distance will exist between the feature vectors of an input pattern and its reference, thereby degrading the recognition performance.

It is natural to consider an appropriate combination of some methods in order to gain a better result. Because different features are obtained by different measurements of the character patterns, it is reasonable to suppose that the features may have different character distributions on their features spaces. These characters troublesome for certain features may be very distinguishable for some other features. Thus, the separability among the character categories will be greatly enhanced if several different features are utilized jointly in recognition. This strategy is commonly adopted for Chinese character recognition. Hagita and Masuda combined stroke distribution method and the direction stroke length distribution method [Hagita and Masuda 1981]. Immediately after this work, research which combined three kinds of features: line direction, crossing count, and background features, was reported [Fujii, et al. 1981]. For preclassification, local line direction can be used with the peripheral area vector [Takahashi 1982]. Most feature combination methods are first applied on handprinted Chinese character. The major problem with the combination schemes is that it is difficult to chose those features which are mutually independent. It is the mutually independence that makes the different features selected sensitive to different disturbances and improves the stability of recognition by means of feature

combination. To cope with such problem, an approach combining four independent features, namely crossing count, stroke proportion, and two peripheral features, has been proposed [Zhang, et al. 1989]. To find out these four features, a correlation analysis of the distance has been made for all possible feature pairs among the four features.

2.3.2 Structural techniques

Chinese characters are patterns which are highly structured. The structure of Chinese characters can be divided into three levels: the whole character level, the parts (radicals) level, and the stroke level. The character level is the highest level while stroke level is the lowest. For two radicals in a character, one radical may be on the left side of the other radical, over the other radical, or surrounded by the other radical. Two strokes inside a radical may be unconnected, or one stroke may contain some connecting points which join that stroke to the other stroke. It is very difficult to use classical statistical approaches to describe such complex pattern structures and relations between subpatterns. It is the structural properties of Chinese characters that make the structural approaches very promising in handwritten and printed CCR. Recently, more efforts have been carried out in this direction. The structural approaches for CCR can be divided into two main streams, namely *grammar approaches* and *graph approaches*.

Grammar method

In the grammar methods, the pattern is represented as a string and a grammar, either context-free or a restricted context sensitive grammar, such as indexed grammar or programmed grammar, which is used to describe the characters [Zhang and Xia 1983; Tai; 1984]. A parser for that particular grammar is built to recognize the pattern. Further development along this line includes stochastic languages [Fu 1982], error correcting parsing, and stochastic error correcting parsing [Lee and Fu 1977]. However, string grammar is still not powerful enough to handle very complex objects like handwritten Chinese characters, and therefore the higher dimensional grammars (tree, plex, or web) are developed. These traditional grammar approaches are still weak in handling noisy or distortion patterns and numerical semantic information [Tsai and 1980]. This shortcoming can be overcome if the attributed grammar approach is used. Considering the characteristics of handwritten Chinese characters and the existing problems, a two-dimensional extended attributed grammar has been proposed [Zhao 1990]. This method carries both overall character shape information and local statistic features of samples, and conducts top-down matching and bottom-up reduction. As the Chinese character set is very large and the grammar describing characters in such a set is quite complicated, it is very difficult to do the parsing. Hence, it is not practical to use grammar approaches for CCR.

Graph methods

Another line of development is to use pattern matching instead of parsing. In this approach, the pattern is usually represented as a relational graph and graph matching is used. In order to incorporate more information into the relation graph, attributed graphs are used to combine the structural and statistical approaches [Tsai and Fu 1979; Shi and Fu 1983]. The attributed graph gives a very flexible representation of structural patterns, especially for handwritten Chinese Characters, in which case, the pattern primitives or vertices of the attributed graph can represent the strokes while the arcs or edges of the attributed graph can represent the relationships between the strokes. In realizing that many natural properties and relations of handwritten Chinese characters are fuzzy, the next step is to include fuzzy attributes in the attributed graph and use this information for further processing. The fuzzy attributed graph for handwritten CCR was proposed [Chan, et al. 1989]. The major drawback of this approach is that the structural properties of Chinese characters are not fully utilized and it is very difficult to organize the model database for efficient searching. Regarding the three structural levels of Chinese characters, the hierarchical attributed graph representation is proposed for handwritten CCR in this thesis. The hierarchical attributed graph represents whole character with its vertices describing the radicals in the character and its arcs describing the spatial relations between the radicals. In the hierarchical attributed graph, a radical attributed graph corresponding to each vertex is used to represent a radical with its vertices describing the strokes and its

edges describing the relations between the strokes. With the HAGR, the huge model database can be organized as a tree with several levels which facilitates accurate and fast searching.

Chapter 3

Preprocessing

3.1 Introduction

Images of input handwritten Chinese characters are obtained by a video camera, and then normalized and transformed into binary images. A binary image is actually a two-dimensional array where each element (pixel) is either 1 or 0. The character pattern consists of those pixels of value 1. Each stroke segment of the character pattern is more than one-pixel thick. Various types of information can be extracted from a binary image by some basic low level operations. The input character pattern in the binary image is skeletonized through a thinning algorithm. Afterwards, the skeleton of the input character is traced to obtain the stroke segments which are then merged to form the strokes of the input character. Geometric information such as the direction and position of strokes can then be extracted. According to the positions of

strokes and the connection relations between strokes, strokes can be grouped into the radicals of the input character. All these operations are considered as preprocessing which provides the local properties of the input character. These properties are then organized for further image analysis and representation.

3.2 Thinning

Thinning is a process by which a binary pattern is transformed into another binary pattern consisting of its skeleton. The major objectives of thinning in pattern recognition and image processing are to reduce data storage and transmission requirements, to reduce the amount of data to be processed, and to facilitate the extraction of features from the pattern.

Many thinning algorithms have been reported [Wakayamas 1982; Lu and Wang; 1985; Pavlidis 1982; Zhang and Suen 1984; Naccache and Shinghal 1984; Zhang and Fu 1984; Plamondon and Suen 1989]. The thinning algorithm described by Zhang and Suen is simple and fast, and can be implemented in parallel, however, the algorithm cannot prevent excessive erosion, so lines or curves that represent the true features of the object tend to be excessively shortened. In our system, the Zhang-Suen algorithm is modified to overcome this erosion problem.

The Zhang-Suen algorithm extracts the skeleton of the character pattern by removing all the edge pixels of the pattern except pixels that belong to the skeleton. In order to preserve the connectivity of the original pattern in the skeleton, iterative

transformations are applied to the binary image of the character pattern and each iteration is divided into two subiterations.

A 3×3 window is used to extract the skeleton. Let n_0 represent the given pixel (i, j) , the eight neighbors inside its window are n_1 to n_8 (see Figure 3.1).

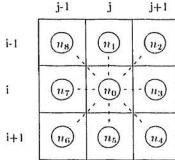


Figure 3.1 Eight neighboring pixels of a pixel n_0

In the first subiteration, according to the values of the eight neighboring pixels, contour n_0 is deleted from the pattern if it satisfies all the following conditions:

$$2 \leq Z(n_0) \leq 6 \quad (3.1)$$

$$N(n_0) = 1 \quad (3.2)$$

$$n_1 \cdot n_3 \cdot n_5 = 0 \quad (3.3)$$

$$n_3 \cdot n_5 \cdot n_7 = 0 \quad (3.4)$$

where $Z(n_0)$ is the number of nonzero neighbors of n_0 , and $N(n_0)$ is the number of "01" patterns in the ordered set n_1, \dots, n_8 .

In the second subiteration, the conditions (3.3) and (3.4) are changed into

$$n_1 \cdot n_3 \cdot n_7 = 0 \quad (3.5)$$

$$n_1 \cdot n_5 \cdot n_7 = 0 \quad (3.6)$$

and the rest remain the same.

By the conditions (3.3) and (3.4) of the first subiteration, the south-east edge pixels and the north-west corner pixels which do not belong to the skeleton are removed. Similarly, the pixel removed by the conditions (3.5) and (3.6) in the second iteration might be a north-west boundary pixel or a south-east corner pixel. The iteration continues until no more pixels can be removed.

It was concluded by Zhang and Suen that:

- By condition (3.1) the endpoints of a skeleton line are preserved.
- Also, condition (3.2) prevents the deletion of those pixels that lie between the endpoints of a skeleton line.

If the algorithm is applied on the pattern in Figure 3.2(a), the pattern would be excessively shortened. This pattern consists of a horizontal section and a diagonal section each of which is two pixel wide. After thinning, the diagonal section is deleted because of the over erosion of the algorithm(see Figure 3.2(b)).

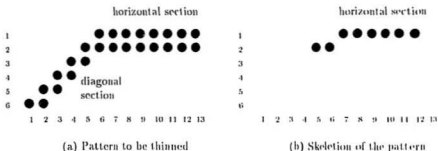


Figure 3.2 Example of over erosion by Zhang-Suen algorithm.

In the first iteration, the end pixel (6,1) is deleted because this pixel satisfies all the conditions of the first subiteration. After pixel (6,1) is deleted, in the second subiteration, the pixel (6,2) is deleted because it satisfies all the conditions for deletion. No other pixels of the diagonal segment can be removed during the first iteration as none of them satisfies all the conditions of the first or second subiteration. In the similar way, pixels (5,2) and (5,3) of the diagonal segment are removed in the second iteration, pixels (4,3) and (4,4) of the diagonal segment are removed in the third iteration, and so on. Until the fifth iteration, no pixel can be further removed from the diagonal segment. Hence, only one pixel (2,5) of the diagonal segment is preserved after thinning.

It is clear that the problem of over-erosion is very severe for the Zhang-Suen algorithm. In an extreme case, when a diagonal segment of two pixels wide is very long, the segment would vanish entirely. This will increase the difficulties in further image analysis and representation of the truly features for character recognition. In

order to avoid the over-erosion problem, a modified algorithm is proposed. In the first subiteration, in addition to the set of original conditions (3.1) to (3.4), a set of alternative conditions is also introduced:

$$n_1 \cdot (n_3 + n_4 + n_7) = 2 \quad (3.7)$$

$$N(n_1) = 2 \quad (3.8)$$

$$Z(n_1) = 4 \quad (3.9)$$

The contour pixel n_0 is deleted if either the set of original conditions is satisfied or the set of alternative conditions is satisfied. The set of alternative conditions is used to guarantee that if the pixel n_0 is on a diagonal segment which is two pixels wide and its neighboring pixel n_1 is not on background, then n_0 is removed. Therefore, the segment at n_1 becomes one pixel wide and the pixel n_1 can be preserved in the following iteration.

Similarly, in the second subiteration, n_0 is removed if either the set of original conditions (3.1), (3.2), (3.5), and (3.6) or the following set of alternative conditions is satisfied:

$$n_5 \cdot (n_3 + n_5 + n_7) = 2 \quad (3.10)$$

$$N(n_1) = 2 \quad (3.11)$$

$$Z(n_1) = 4 \quad (3.12)$$

With the modified algorithm, the over erosion problem is overcome. The comparison is illustrated in Figure 3.3. Figure 3.3(a) is the original pattern. Figure 3.3(b) is

the skeleton of the pattern obtained by the Zhang-Suen algorithm. It is obvious that the diagonal section of the pattern is over shortened. However, the diagonal section is preserved by the modified algorithm(see Figure 3.3(c)).



(a) Original pattern (b) Zhang-Suen algorithm (c) Modified algorithm

Figure 3.3 Comparisons of the thinning algorithms.

3.3 Tracing

After thinning, the skeleton of a character is traced in order to extract the line segments.

Definition 3.1 An end-point "E" in a skeleton has only one neighbor pixel(point) in the skeleton.

Definition 3.2 A joint-point (marked as "+" in a skeleton) has at least three neighbor pixels in the skeleton.

While tracing the skeleton of an input character, all the feature points (joint-points and end-points) are detected. A segment can be extracted by tracing from

one feature point to the other. When a skeleton pixel is visited, it is marked and its coordinates are recorded. By scanning row by row from top to bottom, if an unmarked pixel (called *locating point*) is found, the remaining skeleton connected by this pixel can be traced and marked and the corresponding segments can then be extracted.

Definition 3.3 A *separating point* "S" is a pixel connecting two line segments which have different directions.

The marked segments are first encoded by Freeman's chain code. The codes "A" to "H" are used to indicate eight different directions as shown in Figure 3.4.

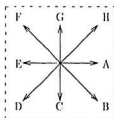


Figure 3.4 Codes and directions for Freeman's chain coding.

Each segment is coded from top to bottom and from left to right. For a segment, coding starts from its one feature point on top or left and ends at another feature point on bottom or right, respectively. The coding of a segment in a loop starts from its locating point "+" and ends at the point "+" along an anti-clockwise direction. Several examples are given in Figure 3.5.

2. Count the number of different kinds of code (denoted as N_d) and the number of codes which are consecutive and identical (denoted as N_s).
3. If N_d reaches three then the scanned substring of the chain code is selected as a line segment.
4. If the selected substring is too short (say, less than four), the first code is neglected. Decrease N_d to less than three, and continue the procedure until N_d reaches three again.
5. If the number of consecutive and identical codes (N_s) is greater than three, then the chain code is selected and the corresponding line segment is extracted.
6. If two line segments are connected to each other, and have the same primitive code, then they are combined and a new line segment is selected to replace them.

For example, the chain code of segment "6" in Figure 3.5(b) is scanned. The first five codes contain two kinds of code, $N_d = 2$. When the 6th code is scanned, the first five codes are selected and the corresponding line segment with the vertical direction is extracted. The procedure continues until the end of the chain code is reached. The loop is represented by four line segments: "DCCCC"(corresponding to the left vertical line segment), "BAAAAA"(corresponding to the bottom horizontal line segment), "HGGGG"(corresponding to the right vertical line segment), and "FEEEEEE"(corresponding to the top horizontal line segment).

3.4 Merging

Two line segments can be merged to form one stroke if they share the same key point(feature point, locating point, or separating point) and have the same orientation. The remaining line segments which cannot be merged remain as individual strokes. For each line segment, the orientation can be determined by:

$$\alpha = \tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \quad (3.13)$$

where (x_1, y_1) and (x_2, y_2) are the two keypoints of the line segment. The orientation of a line segment is:

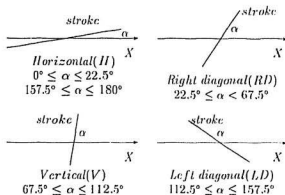


Figure 3.6 Classification of the line segments

- 1) horizontal(H), if $0^\circ \leq \alpha < 22.5^\circ$ or $157.5^\circ \leq \alpha \leq 180^\circ$,
- 2) right diagonal(RD), if $22.5^\circ \leq \alpha < 67.5^\circ$,
- 3) vertical(V), if $67.5^\circ \leq \alpha < 112.5^\circ$,

4) left diagonal(LD), if $112.5^\circ \leq \alpha < 157.5^\circ$.

Some false line segments may occur between joint-points(see Figure 3.7). As their lengths tend to be very short and their directions are very difficult to determine, these segments may create confusion in the stroke merging process. Hence, a process of eliminating such segments (false strokes) is necessary.

One existing method reported in [Chen, et al 1988] is to select a threshold value according to the height (H) of the input character, namely $H/8$. If the length of a segment is smaller than this value then the segment is eliminated. The results obtained by this method are quite unstable, as the threshold value is too inaccurate. The larger the width of the original unthinned pattern, the longer the false stroke becomes. Hence, the false stroke can be longer than the threshold value $H/8$. The maximum circle technique[Liao and Huang 1990] was proposed to decide whether a short line segment between two joint-points is a false stroke. The largest circle centered at each cross-point within the original unthinned binary pattern of the character is formed first. Thus, the size of the largest circle is determined by the width of the original unthinned binary pattern. If the circles centered at both joint-points intersect, the short line segment is a false stroke and should be eliminated. Compared with the first method, the maximum circle technique is more reliable. However, in some cases, it is still very difficult to remove false strokes by maximum circle technique.

For Chinese characters, most false strokes happen when two strokes cross each other. Provided that the width of the unthinned pattern is fixed, the length of a false

stroke is affected by the angle at which the two strokes cross. The smaller the angle, the longer the false stroke is. Based on this observation, an angle-width method is developed for eliminating false strokes. The method takes into account not only the width of the original unthinned pattern but also the angles between strokes. The threshold value is dynamically decided.

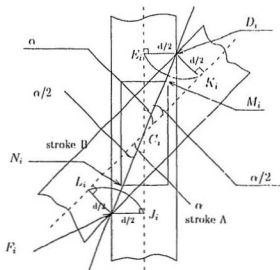


Figure 3.7 False stroke eliminating.

Let d be the width of the unthinned pattern (see Figure 3.7), where the dash lines depict the ideal positions of the skeletons for both strokes. The acute angle between the skeletons of stroke A and stroke B is α . The ideal skeletons of both strokes should meet at point C_i . Stroke A can be thinned from D_i to E_i by a distance of $d/2$ and from F_i to J_i by a distance of $d/2$. Similarly, stroke B can be decreased from D_i to K_i by a distance of $d/2$ and from F_i to L_i also by a distance of $d/2$.

The thinning algorithm shrinks the unthinned pattern by an equal distance from its boundary pixels. The length of $\overline{D_i C_i}$ is longer than $d/2$. The shrinking along the direction of line $\overline{D_i F_i}$ cannot reach the point C_i because the shrinking along the perpendicular direction of line $\overline{D_i F_i}$ reaches line $\overline{D_i F_i}$ first. Thus a false stroke occurs. The length of line $\overline{D_i F_i}$ is $d/\sin(\alpha/2)$. The maximum distance of the shrinking from the point D_i along the direction perpendicular to the direction of line $\overline{E_i J_i}$ or line $\overline{K_i L_i}$ is $d/2$ (from D_i to E_i or K_i). As point D_i is the intersection point of both boundary lines of strokes A and B, the thinning algorithm would shrink the unthinned pattern towards its inside along any direction with the same distance. Therefore, the maximum distance of the shrinking from the point D_i along the direction $\overline{D_i F_i}$ would also be $d/2$ (from D_i to M_i). Similarly, the maximum distance of the shrinking from the point F_i along the direction of line $\overline{F_i D_i}$ would be $d/2$ (from F_i to N_i). The length of the false stroke will not be larger than that of the line $\overline{M_i N_i}$. In this thesis, the length of the line $\overline{M_i N_i}$ is selected as the threshold value T , defined as:

$$T = \frac{d}{\sin(\alpha/2)} - d + \delta l \quad (3.14)$$

where $0 \leq \delta l \leq d$ is a term introduced to compensate for the error due to discretization. In the proposed method, δl can be selected as:

$$\delta l = d - d \sin(\alpha/2) \quad (3.15)$$

Substituting δl in (3.14), we have:

$$T = \frac{d}{\sin(\alpha/2)} - d \sin(\alpha/2). \quad (3.16)$$

In the discrete case, the angle(θ_1) from $\overline{C_i E_i}$ to $\overline{C_i K_i}$ is not equal to the angle(θ_2) from $\overline{C_i L_i}$ to $\overline{C_i J_i}$. Because of the distortion in such a case, α is approximated to be $(\theta_1 + \theta_2)/2$.

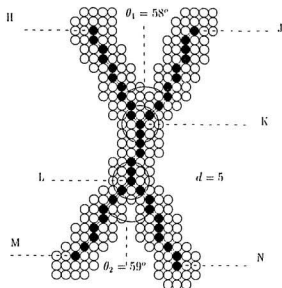


Figure 3.8 A pattern of character with superimposed skeleton which contains a false stroke.

A false stroke between two joint-points is removed if its length is smaller than T . In Figure 3.8, an example is used to illustrate the advantages of this method. The width of the unthinned pattern is 5; the skeleton of the pattern contains a stroke KL ; the length of the false stroke is 7. According to (3.16), we have $T = 7.78$. Stroke KL is detected as a false stroke ($7 < T = 7.78$) and can therefore be removed. The false stroke KL cannot be detected by the maximum circle technique because the largest circles centered at cross points K and L do not overlap.

The following example shows that the angle-width method can also preserve the real strokes between joint-points. In Figure 3.9, the character contains a real stroke BE between two joint points B and E . The length of the stroke BE is 13. According to (3.16), $T = 0.96$. Stroke BE is not a false stroke ($13 > T = 0.96$) and is therefore preserved.

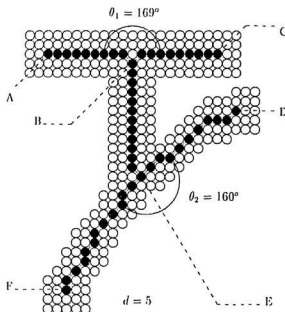


Figure 3.9. A pattern of character with superimposed skeleton which contains real stroke between two cross points.

3.5 Stroke grouping

According to the position of the stroke and the connection between the strokes, the strokes extracted for a character are grouped into component radicals. Suppose that stroke s_1 has a set of m key points (joint-points, end-points, and break-points): $K_1 = \{p_1, \dots, p_m\}$, and stroke s_2 has a set of n key points: $K_2 = \{p'_1, \dots, p'_n\}$, where $p_i = (x_i, y_i)$ and $p'_j = (x'_j, y'_j)$. Two strokes are directly connected with each other if and only if

$$K_1 \cap K_2 \neq \phi. \quad (3.17)$$

Two strokes s_i and s_l are connected with each other, if and only if there is a sequence of strokes, $s_1, s_2, \dots, s_{l-1}, s_l$, such that any two strokes s_i and s_{i+1} are directly connected. All the radicals of the Chinese characters can be classified into three different classes (shown in Figure 3.10):

- 1) Connected radical: each stroke in the radical connects with one of the other strokes in the radical.
- 2) Unconnected radical: each stroke does not connect with any other stroke in the radical.
- 3) Partially connected radical: neither any of above cases.

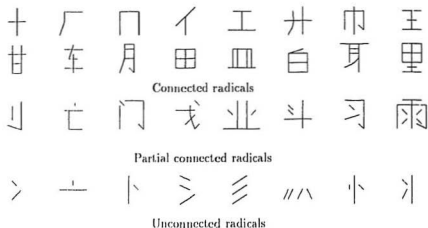


Figure 3.10 Different classes of radicals.

Most radicals of Chinese characters are connected radicals. With connectivity, connected radicals can be easily obtained. Partially connected or unconnected radicals are difficult to determine by their stroke connectivity. However, Chinese characters are block patterns. All strokes of a character are confined to a box area. The strokes of a radical are bounded by a rectangle which is confined to a smaller area inside the box. Based on the connecting and block properties of radicals, connection clustering(C-CT), rectangle clustering(R-CT), binary dividing clustering(BD-CT), and distance measurement clustering(DM-CT) are introduced to obtain the component radicals for handwritten Chinese characters.

3.5.1 Connection clustering

Connection clustering is based on the connection between two strokes. In each cluster, each pair of strokes connect with each other.

3.5.2 Rectangle clustering

A bounded rectangle with a minimum area can be formed to cover a set of connected strokes. Suppose that the coordinates of the lower-left hand corner and the up-right hand corner of the rectangle are (x_1, y_1) and (x_2, y_2) , respectively, and a stroke to be clustered has end points (x, y) and (x', y') .

- When $x_1 < x < x_2$ and $y_1 < y < y_2$ (or $x_1 < x' < x_2$ and $y_1 < y' < y_2$), the stroke is overlapped with the rectangle and the stroke is clustered to the set of the connected strokes.
- When $x_1 \leq x, x' \leq x_2$ and $y_1 \leq y, y' \leq y_2$, the stroke is confined by the rectangle. If a separated stroke is confined by both the rectangles of the two sets of connected strokes, the stroke is clustered in the set which is surrounded by the other set. Figure 3.11 shows a character with two radicals. The dot stroke is clustered in the central set of connected strokes. Otherwise, the separated stroke is clustered in the set of connected strokes with its bounded rectangle confining the separated stroke.

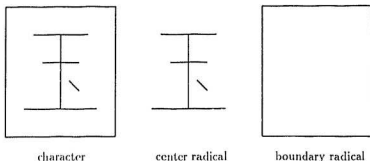


Figure 3.11 Example of rectangle clustering.

Some typical radicals that can be segmented by connection clustering and rectangle clustering are given in Figure 3.12.

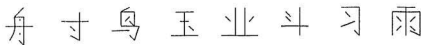


Figure 3.12 Radicals that can be segmented with C-CT and R-CT.

3.5.3 Binary division clustering

Certain radicals always lie in a specific area of the minimum box covering the character. For example, radical 一 is in the top part of the box, radical 丿 is in the bottom part of the box, radical 丿 is in the left part of the box, and radical ㇏ is in the right part of the box. After applying C-CT and R-CT, the bounded box of the character is divided into either top half and bottom half parts, or left half and right half parts. Those unconnected radicals can then be clustered by the part in which the separated strokes are located.

3.5.4 Distance measurement clustering

For a partially connected radical, a separated stroke may neither be confined nor overlapped by bounded rectangles of the connected strokes. By calculating the distances between the end point or the center point of a stroke and the edges of the bounded rectangle of the connected strokes, the stroke can be grouped into the nearest radical.

The algorithm to group the strokes into radicals includes the following four steps:

step 1. Apply connection clustering to form connected radicals and to find connected strokes for partially connected radicals.

step 2. Apply rectangle clustering to classify separated strokes.

step 3. Apply binary division clustering to group unconnected radicals.

step 4. Apply distance measurement clustering to classify remaining isolated strokes.

Figure 3.13 shows how the strokes are grouped by the algorithm. The grouping results after each step are confined by boxes of dashed lines.

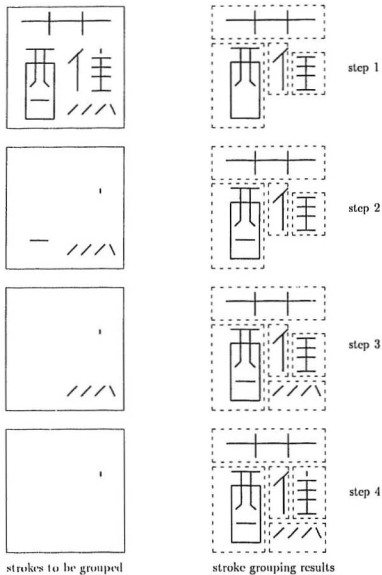


Figure 3.13 Stroke grouping.

Chapter 4

Attributed Graph Representation of Handwritten Chinese Characters

4.1 Introduction

Chinese characters are pictorial patterns consisting of line segments which are called strokes. The concept of strokes defined in this thesis is different from that of conventional strokes which may consist of several single direction line segments. A radical composed of several strokes is the fundamental component of Chinese characters. A Chinese character which consists of one radical is called a single component character. In contrast, a Chinese character which consists of several radicals is called a

compound character.

There are three types of relations between two adjacent radicals: (1) Left-Right, (2) Top-Bottom, and (3) Boundary-Center (see Figure 4.1).

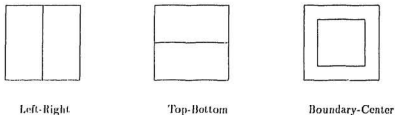


Figure 4.1. Structure relations between two adjacent radicals

In the *Xinhua Dictionary*, more than 7000 Chinese characters are analysed into 150-250 different radicals. The maximum number of radicals within a Chinese character is no more than seven, and over 85% of the characters are composed of no more than three radicals. Several radicals may be structurally combined in a Chinese character to form different patterns as shown in Figure 4.2.

Each radical contains several strokes classified into (1) dot strokes and (2) line strokes with either a horizontal (H), right diagonal (RD), vertical (V), or left diagonal (LD) direction. If two strokes are connected, they may be connected in three basic ways: T-joint (T), L-joint (L), or X (or cross)-joint (X). Hence, a Chinese character can be structurally divided into three levels: whole character, radical, and stroke. These structural features of Chinese characters make the hierarchical attributed graph a very promising and flexible representation for handwritten Chinese characters, as well as providing a direct and simple approach to handwritten Chi-

nese character recognition. It is tolerant to the local variations of the character due to the unstabilities of the stroke length, stroke direction, and stroke connection. In this section, the hierarchical attributed graph representation(HAGR) of handwritten Chinese characters is formally defined for the purpose of character recognition.



Figure 4.2. Structural combinations of radicals.

4.2 Attribute sets and attributed graphs

First, the basic concepts and notations of attributed graphs are introduced.

Definition 4.1 An attribute set is an m-tuple $\langle p_1, \dots, p_i, \dots, p_m \rangle$ where each element in the tuple is an attribute pair. For example, an attribute set describing a line stroke with left diagonal orientation is $\langle (type, line), (orientation, LD) \rangle$.

Definition 4.2 An attributed graph is a graph $G_a = (V_a, E_a)$ where $V_a = \{v_1, \dots, v_p, \dots, v_q, \dots, v_m\}$ is a set of attributed vertices and $E_a = \{\dots, e_{pq}, \dots\}$ is a set of attributed edges/arcs. The edge/arc e_{pq} connects vertices v_p and v_q with an attributed relation.

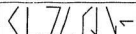

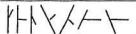



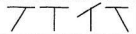

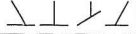

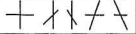

stroke relation	representation
	
	
	
	
	
	

Table 1. Stroke relations.

Definition 4.3 A radical attributed graph $G_r = (V_r, E_r)$ is an attributed graph which represents a radical in a character, where V_r is a set of attributed vertices representing the strokes in the radical and E_r is the set of attributed edges representing the relations between the strokes. Each edge $e_{pq} \in E_r$ joins vertices v_p and v_q without any specification of its direction. Thus the radical attributed graph is an undirected

graph. The different types of relations between pairs of strokes are shown in above table.

As an example, the radical attributed graph of the radical in Figure 4.3 can be represented by:

$G_r = (V_r, E_r)$ where

$$V_r = \{v_1, v_2, v_3, v_4, v_5\}$$

$$E_r = \{e_{12}, e_{13}, e_{14}, e_{23}, e_{24}, e_{25}, e_{34}, e_{35}, e_{45}\}$$

$$v_1 = \langle (type, line), (orientation, H) \rangle$$

$$v_2 = \langle (type, line), (orientation, RD) \rangle$$

$$v_3 = \langle (type, line), (orientation, V) \rangle$$

$$v_4 = \langle (type, line), (orientation, LD) \rangle$$

$$v_5 = \langle (type, line), (orientation, H) \rangle$$

$$e_{12} = \langle (relation, T) \rangle, e_{13} = \langle (relation, X) \rangle$$

$$e_{14} = \langle (relation, T) \rangle, e_{23} = \langle (relation, I) \rangle$$

$$e_{24} = \langle (relation, L) \rangle, e_{31} = \langle (relation, \vdash) \rangle$$

$$e_{45} = \langle (relation, \dashv) \rangle, e_{35} = \langle (relation, X) \rangle$$

$$e_{25} = \langle (relation, \vdash) \rangle$$

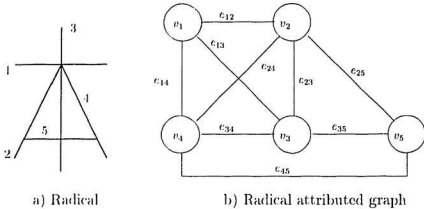


Figure 4.3. Radical attributed graph.

Definition 4.4 A hierarchical attributed graph is an attributed graph $H = (X_h, A_h)$ which represents a Chinese character, where X_h is the set of attributed vertices representing the radicals of the character and A_h is the set of directed attributed arcs which represent the spatial relation of two adjacent radicals. Suppose that e_{st} $\in A_h$ is a directed arc from vertex v_s to vertex v_t . The different spatial relations from V_s to V_t that can be represented by the arc e_{st} are: left-to-right (LR), right-to-left (RL), top-to-bottom (TB), bottom-to-top (BT), boundary-to-center (BC), or center-to-boundary (CB). A complete HAGR of the Chinese character 𠂇 is given in Figure 4.4. This character consists of four radicals: 丿 at the left, ㇏ at the middle-top, 乚 at the right-top, and 𠂇 at the right-bottom. The bottom half portion of the figure shows a HAGR of the character. The windows with round corners represent the four radical attributed graphs of the corresponding radicals.

$H = (V_h, A_h)$ where

$$V_h = \{v_a, v_b, v_c, v_d\}$$

$$A_h = \{e_{ab}, e_{ba}, e_{ad}, e_{da}, e_{bc}, e_{cb}, e_{bd}, e_{db}, e_{cd}, e_{dc}\}$$

$$v_a = \langle \text{number.of.strokes}, 2 \rangle, v_b = \langle \text{number.of.strokes}, 4 \rangle$$

$$v_c = \langle \text{number.of.strokes}, 4 \rangle, v_d = \langle \text{number.of.strokes}, 4 \rangle$$

$$e_{ab} = \langle \text{spatial.relation}, \text{left.to.right} \rangle$$

$$e_{ba} = \langle \text{spatial.relation}, \text{right.to.left} \rangle$$

$$e_{bc} = \langle \text{spatial.relation}, \text{left.to.right} \rangle$$

$$e_{cb} = \langle \text{spatial.relation}, \text{right.to.left} \rangle$$

$$e_{bd} = \langle \text{spatial.relation}, \text{top.to.bottom} \rangle$$

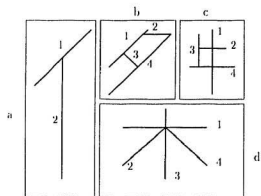
$$e_{db} = \langle \text{spatial.relation}, \text{bottom.to.top} \rangle$$

$$e_{cd} = \langle \text{spatial.relation}, \text{top.to.bottom} \rangle$$

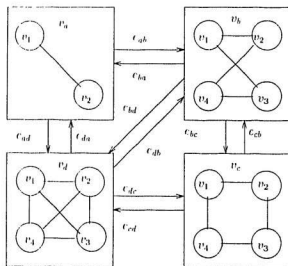
$$e_{dc} = \langle \text{spatial.relation}, \text{bottom.to.top} \rangle$$

$$e_{ad} = \langle \text{spatial.relation}, \text{left.to.right} \rangle$$

$$e_{da} = \langle \text{spatial.relation}, \text{right.to.left} \rangle$$



(1) Candidate character



(2) HAGR for the character

Figure 4.1. A candidate character and its HAGR.

4.3 The data structure of an attributed graph representation

Two data structures can be adopted for the representation of a graph: *adjacency list* and *adjacency matrix* [Hopcroft, et al. 1974], the first one is suitable for large sparse graphs, while the second is suitable for compact graphs. Since the attributed graph of a Chinese character is relatively compact in most cases, the adjacency matrix has been chosen as the data structure of the attributed graph in the present system.

Definition 4.5 An attributed adjacency matrix is the incidence matrix of an attributed graph $G_n = (V_n, E_n)$ of order n with \bar{v}_i as diagonal entries and τ_{ij} as non-diagonal entries. The diagonal entry \bar{v}_i represents the attribute set of vertex v_i and nondiagonal entry τ_{ij} represents that of edge/arcs e_{ij} . If there is no edge between vertices v_i and v_j then τ_{ij} can be set to 0. This is written as:

$$A := [a_{pq}]_{m \times m} \quad (4.1)$$

$$a_{pq} = \begin{cases} 0 & \text{if there is no edge between } v_p \text{ and } v_q \\ \bar{v}_p & \text{if } p = q \\ \tau_{pq} & \text{if } p \neq q, \quad e_{pq} \in E_n \end{cases} \quad (4.2)$$

Definition 4.6 A radical adjacency matrix is an attributed adjacency matrix which represents a radical attributed graph. The diagonal entries of the matrix describe the vertices of the radical attributed graph while the non-diagonal entries describe the

edge of the radical attributed graph. As the radical attributed graph is undirected, its radical adjacency matrix is a symmetric matrix.

The bits of the entries in the matrix describe the attribute set associated with either the vertex v_i or edge e_{ij} . Four orientations (left diagonal, vertical, right diagonal, and horizontal) and two stroke types (line and dot) are used to represent the attribute set of a vertex. Six types of the spatial relations between two strokes are used to represent the attribute set of an edge. For the attribute adjacency matrix of a radical attributed graph, the six rightmost bits of diagonal entry a_{ii} store the attribute set of vertex v_i and the six rightmost bits of nondiagonal entry a_{ij} store the attribute set of edge e_{ij} . The principles to set entry a_{pq} in the adjacency matrix are:

CASE 1. $p = q$, a_{pp} represents the attributed set of vertex v_p .

$a_{pp}(5)$	$a_{pp}(4)$	$a_{pp}(3)$	$a_{pp}(2)$	$a_{pp}(1)$	$a_{pp}(0)$
LD	V	RD	H	•	—

If the attribute value of the orientation of v_p = left diagonal, then $a_{pp}(5) = 1$, else $a_{pp}(5) = 0$. Other bits of the entry a_{pp} can be set similarly, where $a_{pp}(i)$ represents the i th right most bit of the entry a_{pp} in the radical adjacency matrix.

Case 2. $p \neq q$ and $e_{pq} \in E$. a_{pq} represents the attributed set of edge e_{pq} .

$a_{pq}(5)$	$a_{pq}(4)$	$a_{pq}(3)$	$a_{pq}(2)$	$a_{pq}(1)$	$a_{pq}(0)$
X	⊥	⊥	⊥	⊥	L

If the relation of e_{pq} is X, then $a_{pq}(5) = 1$, else $a_{pq}(5) = 0$. Other bits of a_{pq} can

be set similarly, where $a_{pq}(i)$ represents the i th right most bit of the entry a_{pq} in the radical adjacency matrix.

For example, Figure 4.5 illustrates how the attribute set of vertex v_1 and the attribute set of edge e_{12} of radical \nearrow are stored according to the above principles, where $v_1 = \langle (type, line), (orientation, RD) \rangle$, and $e_{12} = \langle (relation, \top) \rangle$.

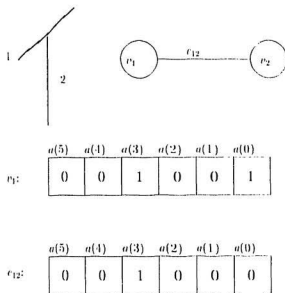


Figure 4.5. Bits of the entries in the adjacency matrix of radical \nearrow .

The radical adjacency matrix of the radical in Figure 4.3 is :

$$A = [a_{ij}] = \begin{pmatrix} 000101 & 001000 & 100000 & 001000 & 000000 \\ 001000 & 001001 & 000100 & 000001 & 000010 \\ 100000 & 000100 & 010001 & 000010 & 100000 \\ 001000 & 000001 & 000010 & 100001 & 000100 \\ 000000 & 000010 & 100000 & 000100 & 000101 \end{pmatrix}$$

There are three advantages to using bits of an entry to represent an attributed set of a vertex or an edge in a radical attributed graph:

- Storage is reduced. Usually, an attribute set of a vertex with k attributes requires k storage units. If a radical graph has n vertices, the corresponding adjacency matrix requires $n \times n \times k$ storage units. However, using the bits of the entry, it only requires $n \times n$ storage units. Since the Chinese character set is very large, the storage size can be significantly reduced.
- The efficiency of graph matching is improved. By using bits of the entry to represent the attributed set, graph matching becomes a comparison of the corresponding bits in the two entries.
- The variation related to the stroke type or orientation is allowed without having to increase the number of models for each character in the database. For example, the fourth stroke of radical 木 in Figure 4.6 can be written as a dot or a line depending on the writing habits or styles of the individual. A simple

way to represent this variation is to set both the 0th and 1st rightmost bits of a_{44} equal to 1.

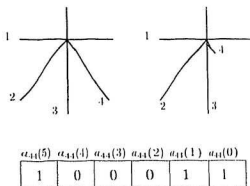



Figure 4.6. Using entry of adjacency matrix

to represent the variations of stroke in the model.

Definition 4.7 A *character adjacency matrix* is an attributed adjacency matrix which represents a hierarchical attributed graph. The diagonal entries of the matrix describe the vertices of the hierarchical attributed graph while the non-diagonal entries describe the directed arcs of the hierarchical attributed graph. As a hierarchical attributed graph is a directed graph, its character adjacency matrix is an anti symmetric matrix.

In the adjacency matrix of a hierarchical attributed graph, the diagonal entry h_{ii} stores the attributed value of the vertex v_i which represents a radical in the character. The attributed value is the number of strokes in the radical. The nondiagonal entry h_{ij} stores the attributed value of arc e_{ij} which represents the spatial relation between two

adjacent radicals, such as left-to-right, right-to-left, top-to-bottom, bottom-to-top, boundary-to-center, and center-to-boundary. For example, the character adjacency matrix of the character  in Figure 4.4 is:

$$C' = [c_{ij}] = \begin{pmatrix} 2 & LR & 0 & LR \\ RL & 4 & LR & TB \\ 0 & RL & 4 & TB \\ RL & BT' & BT' & 4 \end{pmatrix}$$

4.4 Construction of hierarchical attributed graph

After preprocessing, the hierarchical attributed graph for a handwritten Chinese character or its image is constructed in two stages: (1) based on the strokes of each component radical, the radical attributed graph of that radical is constructed, and (2) by considering each component radical as a vertex and the spatial relation between any two adjacent radicals as a directed arc, the hierarchical attributed graph of the character is obtained.

4.4.1 Radical attributed graph construction

The radical attributed graph can be constructed for models and/or images of the character from a set of geometric primitives such as the coordinates of the key points of strokes as well as the types and the directions of the strokes. The attributed values of the strokes and the relation between each pair of strokes can be obtained by measurement. For each radical of the character, the strokes are assigned as attributed

vertices in its radical attributed graph. The type and the direction of the stroke, can then be considered as attributes and their values can be easily determined. The edges of the graph represent the connection relations between pairs of strokes. If two strokes are connected to the same end point, the edge attribute of both strokes is L . If two strokes are connected to the same key point which is not an end point, the edge attribute is X . In all other cases, the edge attribute is T . By the coordinates of end points, and the relative position of both strokes, the edge type attribute (1 , 1 , 1 , or 1) can be determined.

4.4.2 Hierarchical attributed graph construction

In the construction of a hierarchical attributed graph, the radicals of the character are assigned as attributed vertices in the graph. The number of strokes in a radical can be considered as the attribute value of the corresponding vertex of the radical. The arcs of the graph describe the spatial relations between pairs of adjacent radicals. Whether the arc attribute is LR , RL , TB , BT , BC , or CB can be determined by comparing the coordinates of the lower-left hand corners and the upper-right hand corners of the pair of rectangles that cover the radicals.

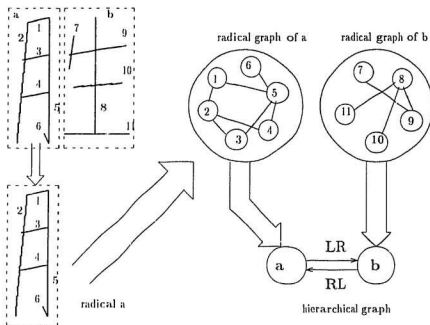


Figure 4.7. Construction of HAGR of a handwritten Chinese character.

The character in Figure 4.7 is used to illustrate the hierarchical attributed graph construction, and the complete HAGR of the character. The character consists of two radicals, 月 and 生. The lower-left portion of Figure 4.7 shows one of them. For each radical there is a radical attributed graph. The upper-right portion shows the radical attributed graphs: one represents the left radical 月 consisting of strokes labelled from 1 to 6, and the other represents the right radical 生 consisting of the strokes 7 to 11. The hierarchical attributed graph is shown on the lower-right of Figure 4.7.

Chapter 5

Recognition of Handwritten Chinese Characters

5.1 Introduction

Once the HAGR of an input candidate handwritten character is constructed, it can then be compared with the model HAGR in the character database. Thus the recognition of a handwritten Chinese character becomes a graph matching problem. In order to provide tolerance to the acceptable variations of the Chinese characters during recognition, a mapping cost function between the adjacency matrices is introduced to the graph matching process. Using this cost function, the HAGR of a candidate character can be matched with its model HAGR which may describe several variations of that character. To speed up and facilitate the matching process, vertex ordering

is applied in the HAGR description for graph matching.

5.2 Radical attributed graph matching


Let $G_c = (V_1, E_1)$ be a candidate radical attributed graph and $G_m = (V_2, E_2)$ be a model attributed graph. A one-to-one correspondence Γ is assigned to $V_1 = \{v_1, \dots, v_n\}$ and $V_2 = \{v'_1, \dots, v'_n\}$, respectively. The mapping cost function is the difference between G_c and G_m under this one-to-one correspondence. Let $A = [a_{ij}]_{n \times n}$ and $B = [b_{ij}]_{n \times n}$ be two adjacency matrices which correspond to the two radical attributed graphs G_c and G_m .

Definition 5.1 The Vertex Mapping Cost (VMC) f_{ij} from the vertex $v_i \in V_1$ to the vertex $v'_j \in V_2$ is:

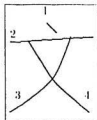
$$f_{ij} = \sum_{k=0}^5 (a_{ii}(k) - b_{jj}(k)a_{ii}(k)) \quad (5.1)$$

where $a_{ii}(k)$ and $b_{jj}(k)$ are the k th rightmost bits of their diagonal entries.

In order to allow for some reasonable variations in stroke orientation, stroke type, and spatial relation between strokes during the recognition process, all variations of the character model are represented by the six rightmost bits of entries in the adjacency matrix. For the candidate character, only one of the variations is represented in the corresponding entry of the radical attributed matrix as described in the previous section. If the stroke(vertex v_i) of a candidate character is one of the variations of the corresponding stroke(vertex v'_j), then the Vertex Mapping Cost f_{ij} is equal to zero.

Figure 5.1 shows an example of the VMC of stroke v_1 in candidate radical 

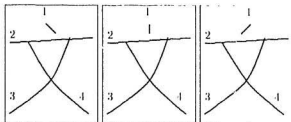
and stroke v'_1 in its model. The top stroke v'_1 in the model is a dot stroke which has three variations: vertical orientation 1 , right diagonal orientation 2 , and left



$v_1 = \langle (type, dot), (orientation, LD) \rangle, a_{11} :$

$a_{11}(5)$	$a_{11}(4)$	$a_{11}(3)$	$a_{11}(2)$	$a_{11}(1)$	$a_{11}(0)$
1	0	0	0	1	0

a) candidate



$v'_1 = \langle (type, dot), (orientation, V/RD/LD) \rangle, b_{11} :$

$b_{11}(5)$	$b_{11}(4)$	$b_{11}(3)$	$b_{11}(2)$	$b_{11}(1)$	$b_{11}(0)$
1	1	1	0	1	0

b) model

Figure 5.1. Vertex matching.

diagonal orientation 3 . Hence, entry b_{11} associated with v'_1 in the adjacency matrix B is set to "111010". The top stroke v_1 in the candidate character has a left

orientation. Entry a_{11} associated with v_1 in adjacency matrix A is set to "100010".

Vertex Mapping Cost of v_1 and v'_1 :

$$\begin{aligned}
 f_{11} &= \sum_{k=0}^5 (a_{11}(k) - b_{11}(k)a_{11}(k)) \\
 &= (a_{11}(0) - b_{11}(0)a_{11}(0)) + (a_{11}(1) - b_{11}(1)a_{11}(1)) + \\
 &\quad (a_{11}(2) - b_{11}(2)a_{11}(2)) + (a_{11}(3) - b_{11}(3)a_{11}(3)) + \\
 &\quad (a_{11}(4) - b_{11}(4)a_{11}(4)) + (a_{11}(5) - b_{11}(5)a_{11}(5)) \\
 &= (0 - 0 \times 0) + (1 - 1 \times 1) + (0 - 0 \times 0) \\
 &\quad + (0 - 1 \times 0) + (0 - 1 \times 0) + (1 - 1 \times 1) + (0 - 0 \times 0) \\
 &= 0
 \end{aligned}$$

Hence, vertex v_1 matches with vertex v'_1 . For the VMC from v_2 to v'_1 , entry a_{22} associated with v_2 is "000101", the cost is:

$$f_{21} = \sum_{k=0}^5 (a_{22}(k) - b_{11}(k)a_{22}(k)) = \overline{(1 - 0 \times 1)} + (0 - 1 \times 0) + \overline{(1 - 0 \times 1)} + \dots = 2$$




This means that vertex v_2 does not match with vertex v'_1 , because the vertical orientation of v_2 is not one of the variations of v'_1 .

Definition 5.2 The Edge Mapping Cost d_{ij} ($i \neq j$) is defined as:

$$d_{ij} = \sum_{k=0}^5 (a_{ij}(k) - b_{ij}(k)a_{ij}(k)) \quad \text{if } c_{ij} \in E_1 \text{ and } c'_{ij} \in E_2 \quad (5.2)$$

where $a_{ij}(k)$ and $b_{ij}(k)$ are the k th rightmost bits of their nondiagonal entries.

Similarly to the VMC, the Edge Mapping Cost d_{ij} is set to zero, if the candidate character belongs to one of the variations in the model.

Figure 5.2 shows an example of EMC of edge e_{12} in candidate radical  and edge e'_{12} in its model. The radical can be written as  or . Therefore, there are two variations for the relation between the strokes in the radical: \vdash and \lrcorner .



$$e_{12} = \langle relation, \vdash \rangle, a_{12} :$$

$a_{12}(5)$	$a_{12}(4)$	$a_{12}(3)$	$a_{12}(2)$	$a_{12}(1)$	$a_{12}(0)$
0	0	0	0	1	0

a) candidate



$$e'_{12} = \langle relation, \lrcorner \rangle, b_{12} :$$

$b_{12}(5)$	$b_{12}(4)$	$b_{12}(3)$	$b_{12}(2)$	$b_{12}(1)$	$b_{12}(0)$
0	0	0	0	1	1

b) model

Figure 5.2. Edge matching.

$$EMC : d_{12} = \sum_{k=0}^5 (a_{12}(k) - b_{12}(k)a_{12}(k)) = 0$$

The EMC of edge e_{12} and edge e'_{12} is zero, therefore edge e_{12} matches with edge

e'_{12} .

Proposition 5.1 Vertex Mapping Cost f_u and Edge Mapping Cost d_{ij} are always greater than or equal to zero.

Proof: For $0 \leq k \leq 5$, both $a_u(k)$ and b_u can only have values 1 or 0.

If $a_u(k) = 0$, $a_u(k) - b_u(k)a_u(k) = 0 - b_u(k) \times 0 = 0$.

If $a_u(k) = 1$, $a_u(k) - b_u(k)a_u(k) = 1 - b_u(k) \times 1 \geq 0$.

So for $0 \leq k \leq 5$, $a_u(k) - b_u(k)a_u(k) \geq 0$.

Hence, $f_u = \sum_{k=0}^5 a_u(k) - b_u(k)a_u(k) \geq 0$.

In the same way, $d_{ij} \geq 0$ can be proved.

Definition 5.3 The Matrix Mapping Cost from adjacency matrix A to adjacency matrix B is defined as

$$MMC(A, B) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} \quad (5.3)$$

where c_{ij} is given by

$$c_{ij} = \begin{cases} f_{ij} & \text{if } i = j \\ d_{ij} & \text{if } i \neq j, c_{ij} \in E_1 \text{ and } c'_{ij} \in E_2 \\ \sum_{k=0}^5 a_{ij}(k) & \text{if } i \neq j, c_{ij} \in E_1 \text{ and } c'_{ij} \notin E_2 \\ 0 & \text{otherwise.} \end{cases} \quad (5.4)$$

Proposition 5.2 c_{ij} can be uniquely represented by

$$c_{ij} = \sum_{k=0}^5 (a_{ij}(k) - b_{ij}(k)a_{ij}(k)) \quad (5.5)$$

for all $1 \leq i, j \leq n$.

Proof: if $i = j$:

$$c_{ij} = c_{ii} = f_{ii} = \sum_{k=0}^5 (a_{ii}(k) - b_{ii}(k)a_{ii}(k)).$$

If $i \neq j$, $c_{ij} \in E_1$ and $c'_{ij} \in E_2$:

$$c_{ij} = d_{ij} = \sum_{k=0}^5 (a_{ij}(k) - b_{ij}(k)a_{ii}(k)).$$

if $i \neq j$, $c_{ij} \in E_1$ and $c'_{ij} \notin E_2$, then $b_{ij}(k) = 0$ for $0 \leq k \leq 5$,

$$\sum_{k=0}^5 (a_{ij}(k) - b_{ij}(k)a_{ij}(k)) = \sum_{k=0}^5 (a_{ij}(k) - 0 \times a_{ij}(k)) = \sum_{k=0}^5 a_{ij}(k) = c_{ij}.$$

If $i \neq j$, $c_{ij} \notin E_1$, $a_{ij}(k) = 0$ for $0 \leq k \leq 5$,

$$\sum_{k=0}^5 (a_{ij}(k) - b_{ij}(k)a_{ij}(k)) = \sum_{k=0}^5 (0 - b_{ij}(k) \times 0) = 0 = c_{ij}.$$

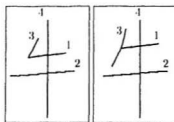
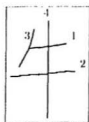
Hence, c_{ij} can be uniquely represented by:

$$c_{ij} = \sum_{k=0}^5 (a_{ij}(k) - b_{ij}(k)a_{ij}(k))$$

for all $1 \leq i, j \leq n$.

In Figure 5.3, the attributed graph of a candidate and that of a model are represented by two matrices. The Matrix Mapping Cost, $MMC(A, B)$, can be used to measure the difference between these matrices.

$$\begin{aligned} c_{11} &= \sum_{k=0}^5 (a_{11}(k) - b_{11}(k)a_{11}(k)) \\ &= (1 - 1 \times 1) + (0 - 0 \times 0) + (1 - 1 \times 1) + \\ &= (0 - 0 \times 0) + (0 - 0 \times 0) + (0 - 0 \times 0) \\ &= 0 \end{aligned}$$



$$G_1 = \{V_1, E_1\}$$

$$V_1 = \{v_1, v_2, v_3, v_4\}$$

$$E_1 = \{e_{12}, e_{13}, e_{14}, e_{23}\}$$

$$v_1 = \langle \{type, line\}, \{orientation, H\} \rangle$$

$$v_2 = \langle \{type, line\}, \{orientation, H\} \rangle$$

$$v_3 = \langle \{type, line\}, \{orientation, HD\} \rangle$$

$$v_4 = \langle \{type, line\}, \{orientation, V\} \rangle$$

$$e_{12} = \langle \{relation, \vdash\} \rangle$$

$$e_{14} = \langle \{relation, X\} \rangle$$

$$e_{24} = \langle \{relation, X\} \rangle$$

$$A = [a_{ij}] :$$

$$\begin{pmatrix} 000101 & 000000 & 000010 & 100000 \\ 000000 & 000101 & 000000 & 100000 \\ 000010 & 000000 & 001001 & 000000 \\ 100000 & 100000 & 000000 & 010001 \end{pmatrix}$$

candidate radical

$$G_m = \{V_2, E_2\}$$

$$V_2 = \{v'_1, v'_2, v'_3, v'_4\}$$

$$E_2 = \{e'_{12}, e'_{13}, e'_{14}, e'_{24}\}$$

$$v'_1 = \langle \{type, line\}, \{orientation, H\} \rangle$$

$$v'_2 = \langle \{type, line\}, \{orientation, H\} \rangle$$

$$v'_3 = \langle \{type, line\}, \{orientation, RD\} \rangle$$

$$v'_4 = \langle \{type, line\}, \{orientation, V\} \rangle$$

$$e'_{12} = \langle \{relation, \vdash / L\} \rangle$$

$$e'_{14} = \langle \{relation, X\} \rangle$$

$$e'_{24} = \langle \{relation, \nabla\} \rangle$$

$$B = [b_{ij}] :$$

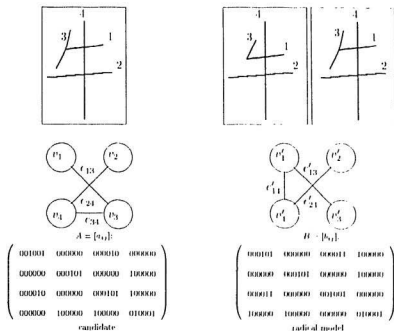
$$\begin{pmatrix} 0000101 & 0000000 & 0000011 & 0100000 \\ 0000000 & 0000101 & 0000000 & 0100000 \\ 0000011 & 0000000 & 0001001 & 0000000 \\ 0100000 & 0100000 & 0000000 & 0010001 \end{pmatrix}$$

radicalmodel

Figure 5.3. Adjacency matrices A and B .




In the same way, $c_{ij} = 0$ for $0 \leq i, j \leq 5$. Hence:

$$MMC(A, B) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} = 0$$



$$MMC(A, B) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} = 4$$

Figure 5.4. MMC of A and B after exchanging the order of v_1 and v_3 .

In the above example, the candidate radical  is one of the variations of the model  and . Hence, $MMC(A, B)$ is equal to zero. However, if the vertices in the graph of the candidate radical are ordered differently, $MMC(A, B)$ cannot be equal to zero. For instance, exchanging the order of stroke v_1 with stroke

v_3 will cause the entry in the 3rd row and column to switch with that in the 1st row and 1st column in A (see Figure 5.4).

Therefore, the vertex order in graph G_c should be arranged properly. To order the vertices, an operation called permutation of the adjacency matrix A is introduced. The MMC and permutation of adjacency matrix can be used to find a match between G_c and G_m .

Definition 5.4 A Primitive row (column) operation of adjacency matrix A exchanges any two rows (columns) of A . A Permutation of adjacency matrix A exchanges the i th and j th rows after exchanging the i th and j th columns in the matrix.

Definition 5.5 If a limited number of permutations have been performed on adjacency matrix A , and $MMC(A, B)$ has reduced to zero, the adjacency matrix A is said to be compatible with adjacency matrix B .

Definition 5.6 Radical attributed graph G_c is said to match radical attributed graph G_m if and only if matrix A is compatible with matrix B .

By using the adjacency matrices A and B to represent the graphs G_c and G_m , finding a match from G_c to G_m determines whether A is compatible with B . Thus certain permutations should be performed on A to see if $MMC(A, B)$ is equal to zero after such permutations. As the bitwise technique is used to represent the attribute set of the vertices and edges in the adjacency matrices, the concept of unit matrix and operations on the adjacency matrices is different from those of traditional matrices which contain numbers as their entries. It is necessary to give new definitions for the

unit matrix and the operations on the matrices.

Definition 5.7 The unit matrix is an adjacency matrix with 111111 as its diagonal entries and 000000 as its nondiagonal entries.

Definition 5.8 The matrices obtained by performing the primitive row or column operation once on the unit matrix is called primitive matrices.

For example, by performing the row operation on the 2nd and 3rd rows of the unit matrix, primitive matrix P_{23} will be obtained as follows:

$$\begin{pmatrix} 111111 & 000000 & 000000 & 000000 \\ 000000 & 111111 & 000000 & 000000 \\ 000000 & 000000 & 111111 & 000000 \\ 000000 & 000000 & 000000 & 111111 \end{pmatrix}$$

unit matrix

$$\begin{pmatrix} 111111 & 000000 & 000000 & 000000 \\ 000000 & 000000 & 111111 & 000000 \\ 000000 & 111111 & 000000 & 000000 \\ 000000 & 000000 & 000000 & 111111 \end{pmatrix}$$

primitive matrix P_{23}

Definition 5.9 The multiplication \star of two adjacency matrices $F = [f_{ij}]_{n \times n}$ and $L = [l_{ij}]_{n \times n}$ is defined as:

$$F \star L = \left[\sum_{k=1}^n (f_{ik} \oslash l_{kj}) \right]_{n \times n} \quad (5.6)$$

where $\sum_{t=1}^n x_t = x_1 \odot x_2 \odot \dots \odot x_n$, \odot is the “bitwise and” operator and \oplus is the “bitwise or” operator. For example, if $x = 010111$ and $y = 110001$, then $x \odot y = 010001$ and $x \oplus y = 110111$.


Proposition 5.3 A primitive row(column) operation on matrix A is to multiply the matrix A on the left(right) hand side with a primitive matrix. A permutation on A is to multiply A on both sides with a primitive matrix.

Proposition 5.4 The adjacency matrix A is compatible with the adjacency matrix B if and only if there exists primitive matrices P_1, \dots, P_l such that

$$MMC(\overbrace{P_1 * P_{l-1} * \dots * P_1 * A * P_1 * P_2 * \dots * P_l}^{A'}, B) = 0 \quad (5.7)$$

In other words, if a set of primitive matrices is used to perform the respective permutations on A and $MMC(A', B)$ is equal to zero, then A is compatible with B . Furthermore from Definition 5.6, G_c matches with G_m .

5.3 Example: finding a radical graph match

Figure 5.5 shows the attributed radical graph G_c of branch  and the attributed graph G_m of its model. Their adjacency matrices are A and B respectively:

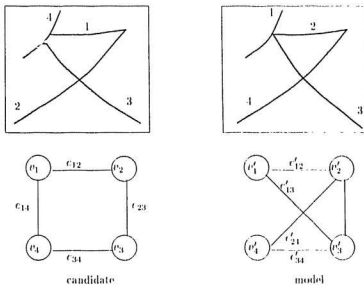


Figure 5.5. Radical  and its model.

$$A = \begin{pmatrix} 000101 & 000001 & 000000 & 000010 \\ 000001 & 001001 & 100000 & 000000 \\ 000000 & 100000 & 100001 & 000010 \\ 000010 & 000000 & 000010 & 001001 \end{pmatrix},$$

$$B = \begin{pmatrix} 001001 & 000010 & 000010 & 000000 \\ 000010 & 000101 & 000001 & 000001 \\ 000010 & 000001 & 100001 & 100000 \\ 000000 & 000001 & 100000 & 001001 \end{pmatrix}.$$

$$MMC(A, B) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=0}^5 (a_{ij}(k) - b_{ij}(k)a_{ij}(k)) = 12 \geq 0.$$

In the first iteration, primitive matrix P_1 is selected as follows:

$$P_1 = \begin{pmatrix} 000000 & 000000 & 000000 & 111111 \\ 000000 & 111111 & 000000 & 000000 \\ 000000 & 000000 & 111111 & 000000 \\ 111111 & 000000 & 000000 & 000000 \end{pmatrix},$$

$$P_1 + A + P_1 = \begin{pmatrix} 001001 & 000000 & 000010 & 000010 \\ 000000 & 001001 & 100000 & 000001 \\ 000010 & 100000 & 100001 & 000000 \\ 000010 & 000001 & 000000 & 000101 \end{pmatrix},$$

$$MMC(P_1 + A + P_1, B) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} = 6.$$

In the second iteration, primitive matrix P_2 is selected as follows:

$$P_2 = \begin{pmatrix} 111111 & 000000 & 000000 & 000000 \\ 000000 & 000000 & 000000 & 111111 \\ 000000 & 000000 & 111111 & 000000 \\ 000000 & 111111 & 000000 & 000000 \end{pmatrix},$$

$$P_2 * P_1 * A * P_1 * P_2 = \begin{pmatrix} 001001 & 000010 & 000010 & 000000 \\ 000010 & 000101 & 000000 & 000001 \\ 000010 & 000000 & 100001 & 100000 \\ 000000 & 000001 & 100000 & 001001 \end{pmatrix},$$

$$MMC(P_2 * P_1 * A * P_1 * P_2, B) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} = 0.$$

So matrix A is compatible with matrix B and G_c matches with G_m .

5.4 Ordering the vertices in a radical graph

In order to find a match between G_c and G_m , in the worst case all possible sequences of permutations on A have to be examined to determine if a sequence of permutations with corresponding primitive matrices P_1, \dots, P_l will make $MMC(P_l * \dots * P_1 * A * P_1 \dots P_l, B) = 0$. Unfortunately this is a combinatorial problem. Several methods have been proposed to overcome this problem. One approach is to manually select the control vertices [S. W. Lee and J. M. Kim 1989]. However, for the recognition of handwritten Chinese characters, selecting the control vertices is very difficult (if not impossible) because the positions of the vertices vary with the different writing styles for the same character.

In our approach, the geometric information of the strokes of a radical is used to order the vertices in the radical attributed graph. This reduces significantly the time needed to find the match by performing the permutations on A .

For a stroke, equation $Ax + By + C = 0$ can be determined by the coordinates of its two end-points. Suppose that the equation for stroke a with two end-points (x_{a1}, y_{a1}) and (x_{a2}, y_{a2}) is $A_ax + B_ay + C_a = 0$, and the equation for stroke b with two end-points (x_{b1}, y_{b1}) and (x_{b2}, y_{b2}) is $A_bx + B_by + C_b = 0$, then the four rules to determine the relative order of these two strokes are:

Rule 1. The projections of strokes a and b on the X or Y axis do not overlap.

If $\min\{y_{a1}, y_{a2}\} \geq \max\{y_{b1}, y_{b2}\}$, then stroke a is ordered before stroke b . If $\max\{x_{a1}, x_{a2}\} \leq \min\{x_{b1}, x_{b2}\}$, then stroke a is ordered before b (See Figure 13).

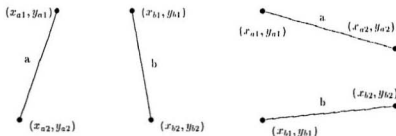


Figure 5.6. Example for ordering strokes a and b using Rule 1.

Rule 2. Strokes a and b do not cross each other and their projections on the X axis overlap. The two end-points (x_{b1}, y_{b1}) and (x_{b2}, y_{b2}) of the stroke b lie on the opposite sides of stroke a (see Figure 5.7). This can be mathematically represented as: $(A_ax_{b1} + B_ay_{b1} + C_a)(A_ax_{b2} + B_ay_{b2} + C_a) < 0$. If $A_bx_{a1} + B_by_{a1} + C_b > 0$ and $A_bx_{a2} + B_by_{a2} + C_b > 0$, then the stroke a is ordered before the stroke b as illustrated on the left hand side of Figure 5.7. However, if

$A_ax_{a1} + B_ay_{a1} + C_b < 0$ and $A_bx_{a2} + B_by_{a2} + C_b < 0$, then the stroke a is ordered after the stroke b as shown on the right hand side of Figure 5.7.

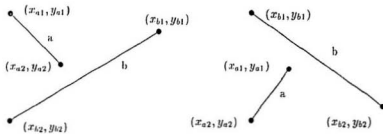


Figure 5.7. Example for ordering strokes a and b with Rule 2.

Rule 3. Stroke a and stroke b intersect each other. The strokes are ordered according to their orientations: H , RD , V and LD . For example, the stroke a is horizontal(H) and the stroke b is right diagonal(RD), then a is ordered before b (See Figure 5.8). If the code of stroke a is the same as that of stroke b and the inclined angle of stroke b is greater than that of stroke a , then stroke a is ordered before stroke b (See Figure 5.8).

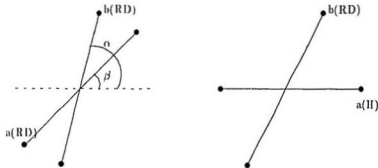


Figure 5.8. Example for ordering strokes a and b with Rule 3.

Rule 4. Stroke a and stroke b can not be ordered by rules 1-3. If $A_b x_{a1} + B_b y_{a1} + C_b > 0$ and $A_b x_{a2} + B_b y_{a2} + C_b > 0$, then a is ordered before b (see Figure 5.9).

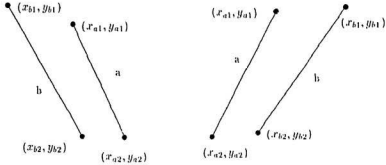


Figure 5.9. Example for ordering strokes a and b with Rule 4.

Applying the ordering rules, a set of strokes in a radical can be sorted to create an ordered list by the following procedure:

1. Sort the strokes of the radical into a non-increasing sequence T according to the vertical coordinates of the upper end-points of the strokes.
2. Initialize an empty list L for the ordered list.
3. While ($T \neq \text{empty}$) do
 - a) remove the first stroke f from T and place it at the end of the list L ,
 - b) use the ordering rules to insert stroke f into an appropriate place in L .

To illustrate the stroke ordering algorithm, an example is given in Figure 5.10. All strokes are sorted into a non-increasing sequence T ($T = \{T_1, T_2, \dots, T_6\}$). Then

an ordered list L is derived from T by applying proper ordering rules. Stroke T_1 is removed from T and put into L first. With rule 1, stroke T_2 should be ordered before stroke T_1 and thus is inserted in L before stroke T_1 . According to rule 2, stroke T_3 is after stroke T_2 and stroke T_4 is after T_3 . Again with rule 1, T_5 is after T_4 . At last, T_6 is inserted before T_4 and T_5 , but after T_3 by applying the rule 3 and rule 1 respectively.

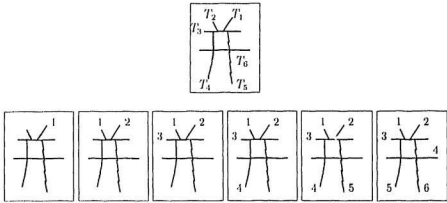


Figure 5.10. Example for ordering strokes in a radical.

5.5 Hierarchical attributed graph matching

Let $H_c = (X_1, A_1)$ be a candidate hierarchical attributed graph and $H_m = (X_2, A_2)$ be a model attributed graph respectively. A one-to-one correspondence Γ is assigned to $X_1 = \{v_1, \dots, v_n\}$ and $X_2 = \{v'_1, \dots, v'_n\}$. Let G_i and G'_i be the radical graphs corresponding to vertices v_i and v'_i . A_i and B_i represent the radical adjacency matrices of G_i and G'_i respectively. The character mapping cost function is the difference

between H_c and H_m under the one-to-one correspondence Γ . Let $L = [l_{ij}]_{n \times n}$ and $K = [k_{ij}]_{n \times n}$ be two character adjacency matrices which correspond to the two hierarchical attributed graphs H_c and H_m .

Definition 5.10 The Character Vertex Mapping Cost F_{ij} from the vertex $v_i \in X_1$ to the vertex $v'_j \in X_2$ is:

$$F_{ij} = \begin{cases} MMC(A_i, B_j) & \text{if } \text{order}(G_i) = \text{order}(G'_j) \\ |\text{order}(G_i) - \text{order}(G'_j)| & \text{otherwise.} \end{cases} \quad (5.8)$$

When radical graph G_i matches with graph G'_j , the Character Vertex Mapping Cost F_{ij} is equal to zero. Otherwise, F_{ij} is not equal to zero.

Definition 5.11 The Arr Mapping Cost D_{ij} is defined as:

$$D_{ij} = \begin{cases} 0, & \text{if } l_{ij} = k_{ij} \\ 1, & \text{otherwise.} \end{cases} \quad (5.9)$$

Definition 5.12 The Character Matrix Mapping Cost from character adjacency matrix L to character adjacency matrix K is defined as:

$$CMC(L, K) = \sum_{i=1}^n \sum_{j=1}^n m_{ij} \quad (5.10)$$

where m_{ij} is given by:

$$m_{ij} = \begin{cases} F_{ij} & \text{if } i = j \\ D_{ij} & \text{otherwise.} \end{cases} \quad (5.11)$$

The Character Matrix Mapping Cost, $CMC(L, K)$, can be used to measure the difference between these two hierarchical attributed graphs H_c and H_m . If $CMC(L, K) = 0$, H_c matches with H_m . Similarly, the vertices in the graph H_c should be

arranged properly. In order to avoid the combinatorial problem during the process of matching, the relative position of the radicals is used to order the vertices in the hierarchical attributed graph. Two radicals in a character can be ordered according to the following principles.

Principle 1. If one radical is on the left hand side of the other one, but is neither over nor under the other, then the left radical is ordered in front of the right.

Principle 2. If one radical is over the other, but is neither on the left hand side nor on the right hand side of the other, the top radical is ordered in front of the bottom.

Principle 3. If one radical is included or surrounded by the other, then the included or surrounded radical is ordered in front of the other.

Definition 5.13 For the hierarchical attributed graph H_c of a candidate handwritten character, and the hierarchical attributed graph H_m of a model, H_c is said to match with H_m if their corresponding Character Matrix Mapping Cost is equal to zero.

Hierarchical attributed representation and graph matching offer an efficient method for Chinese character recognition. With a matching procedure, characters can be recognized if a match is found between the HAGR of the character and a stored model. The character recognition procedure is briefly summarized as follows:

- Preprocessing.

- Construct all the radical attributed graphs for all the radicals of the input character.
- Construct the hierarchical attributed graph H_i for the input character.
- Search the database to find a match between H_i and H_m . If a match is found, the character has been recognized.
- If no match is found, the input character is rejected.

Chapter 6

Model Database Organized by A Heterogeneous Multi-way Tree

6.1 Introduction

In most existing recognition methods, a character model database is required by the recognition system. As the number of Chinese characters is very large (about 7,000 characters in daily use[Tai and Liu 1990]), the model database could be very huge. The recognition process can be very tedious and difficult because searching such a large model database to find a match of the input character is time consuming and possibly inaccurate, if the database is not well organized. Therefore, it is necessary to develop an effective method to organize and search the model database.

One simple way is to organize all the models in a linear list. If the list is searched

sequentially, $O(n)$ comparisons are needed. In order to improve the search of a linear list, a “multistage approach” has been proposed [Zhang, et al. 1989]. Searching the list is divided into several stages. In the first stage, the candidate models for an input character are selected by some “rough” features of the input character. In the following stages, the selected models are reselected according to some “finer” features. In the last stage of recognition, the input character is assigned to a model within the selected candidate set by the method of distance measurement. However, this method has some disadvantages: e.g. (1) it can easily assign a wrong model to the input character because the mutually similar models of the different characters are usually gathered in the selected set; and (2) as the selected candidate set has to be rearranged in a linear list and the whole list has to be used to reselect the candidate models, a large amount of time is still required.

In this chapter, a heterogeneous multi-way tree approach is introduced. It makes use of both the structural and statistical information of Chinese characters for the purpose of organizing the models in the database. With this approach, a fast and accurate searching algorithm has also been developed for the recognition of handwritten Chinese characters.

6.2 Heterogenous multi-way tree organization

Using the radical alphabet, a means of grouping characters according to their radicals, is an effective way to organize Chinese characters in the Xinhua Dictionary. In this

alphabet, each group is partitioned into several subgroups which contain the same number of strokes. Since the HAGR of a handwritten Chinese character includes spatial relations between radicals, the number of strokes in each radical, relations between strokes, the direction of a stroke, and the type(dot or line) of a stroke, it is very convenient to use the radical alphabet organization similar to the Xinhua dictionary to divide the set of models into disjoint subsets. Furthermore, a tree structure can be used to implement such a division. Hence, searching for a model to match an input character in the database is divided into a number of simple and local decisions at different levels of the tree. Therefore, the subtrees which do not contain the model for the input character can be pruned at an early stage to minimize the searching time. Furthermore, special strategies such as bit-wise representation, radical linked list, and compression vector, have also been applied so that the heterogeneous multi-way tree does not occupy too much memory.

6.2.1 Basic concepts for the heterogeneous multi-way tree

Definition 6.1 A structure adjacency matrix(SAM) $S = [s_{ij}]_{n \times n}$ of a HAGR(or character) is a matrix such that:

$$s_{ij} = \begin{cases} 0 & \text{if } i = j \\ a_{ij} & \text{relation attributes} \end{cases},$$

It is easy to see that characters with identical SAMs would have (1) the same number of radicals and (2) the same spatial relations between radicals. For example,

characters $\begin{array}{c} \diagup \\ \text{十} \\ \diagdown \end{array}$ and $\begin{array}{c} \diagup \\ \text{木} \\ \diagdown \end{array}$ have the same *SAM*, as indicated;

$$S = [s_{ij}]_{3 \times 3} = \begin{pmatrix} 0 & LR & TB \\ RL & 0 & TB \\ BT & TB & 0 \end{pmatrix}.$$

Both of these characters have three radicals, with relation(LR) between radicals $\begin{array}{c} \diagup \\ \text{丿} \end{array}$ and $\begin{array}{c} \diagdown \\ \text{乚} \end{array}$ the same as that between radicals $\begin{array}{c} \text{丨} \\ \text{丨} \end{array}$ and $\begin{array}{c} \text{丨} \\ \text{丨} \end{array}$, and so on.

A heterogeneous tree is a tree composed of different types of nodes. The heterogeneous multi-way tree contains six types of nodes: root node, radical number node, configuration node, combination node, radical node, and character node.

Definition 6.2 A radical-number node is a node with two attributes: radical-number and pointers, where attribute radical-number describes the number of radicals in a character and the pointers point to nodes in the next level of the tree.

Definition 6.3 An index array A for a character with N radicals is an N dimensional array. The element $A[q_1, \dots, q_N]$ of the array corresponds to the character which has q_1 strokes in the 1st radical, \dots , q_n strokes in the i th radical, and so on.

Definition 6.4 A configuration node is a node with two attributes: a structure adjacency matrix (*SAM*) and an index array which contains a set of pointers pointing to combination nodes. The dimension of the index array is N if the size of *SAM* is $N \times N$.

Definition 6.5 The stroke number combination(SNC) of a character(HAGR) is an N -tuple $[q_1, q_2, \dots, q_N]$ where q_i ($0 < i \leq N$) is the number of strokes possessed by the

i th radical of the character. For example, SNC of character  is [2, 4 2].






Definition 6.6 A combination node is a node with two attributes: a stroke number combination (SNC) and a set of pointers pointing to radical nodes or character nodes.

Definition 6.7 A radical node is a node with two attributes: a radical adjacency matrix (RAM) and a set of pointers to radical nodes or character nodes.

Definition 6.8 A character node is a node with two attributes: a RAM and an integer ch_code, the character code, which uniquely identifies a character.

6.2.2 Model database organized by multi-way tree

The model database is considered as a universal set. The universal set can be decomposed into subsets by features of characters such as the radical number(RN), the structure adjacency matrix(SAM), the stroke number combination(SNC), and the radicals which are represented by radical adjacency matrices(RAM). The subsets at each level are disjoint and divided into smaller disjoint subsets at the next lower level.

The universal set is first partitioned into subsets called *parts*. All models in a part have the same RN . For example, characters  and  are in the same *part* with radical number $RN = 2$, where character  belongs to the *part* with radical number $RN = 3$. Each *part* is also divided into disjoint subsets called *sections* according to SAM . For example, the characters  and  are placed in the same section because they have the same SAM . The characters in each *section* are classified by SNC into different disjoint sets called *groups*. Each *group*

is then split into disjoint subgroups which are called "same 1-radical groups" ($S1G$ s) according to the first radical in the characters. For example, 拊 and 拊 have the same first radical 扌 and belong to 扌- $S1G$; 林 and 林 belong to 木- $S1G$. Each $S1G$ can be further divided into disjoint subsubgroups defined as "same 2-radical groups" ($S2G$ s), if the characters in the $S1G$ have more than one radical. The first radicals for all the characters in a $S2G$ are the same and the second radicals for all the characters in the $S2G$ are also the same. For example, the characters, 拊, 拊, and 拊 are in 扌- $S1G$. 拊 and 拊 are in 扌- $S2G$ while 拊 is in 扌- $S2G$. The process of the subdivision can be carried out such that a SiG is subdivided into disjoint subgroups called $S(i+1)G$ if the characters in the SiG have more than i radicals. In each $S(i+1)G$, all characters have the same first, second, ..., $i+1$ radicals.

The hierarchy of the subsets obtained by the subdivision can be represented by a heterogeneous multi-way tree. The root (inverted-triangle node) of the tree represents the universal set. Each *part* is represented by a radical-number node(oval node) at the second level of the tree. The dotted lines show the correspondence between the nodes in the tree and the parts in the universal set. The key is an integer representing the *RN* of the part(see Figure 6.1). The database(universal set) is divided into several *parts*: p_1, \dots, p_m . For example, the characters 丑, 丑, and 丑, each with one radical, belong to part p_1 , whereas the characters 𠂔 and 𠂔 have two radicals and belong to part p_2 . Likewise the characters 𠂔 and 𠂔 have three

radicals and belong to part p_3 and so on.

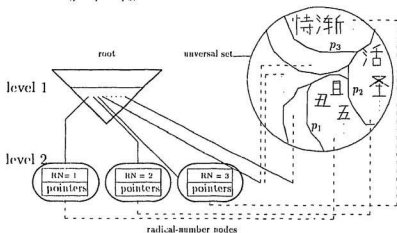


Figure 6.1. Tree representing the partition of universal set.

A *section* is represented by a configuration node (circular node) at the third level of the tree. The *SAM* of a section is stored as the key of the configuration node. For example, part p_3 containing the characters with the same three radicals, is divided into several *sections*: s_1, \dots, s_h (see Figure 6.2). Here, S'_1 is the SAM of characters with spatial structure $\begin{bmatrix} \square & \square \\ \square & \square \end{bmatrix}$, and S'_2 is the SAM of characters with spatial structure $\begin{bmatrix} \square & \square \\ \square & \square \end{bmatrix}$, such that,

$$S'_1 = \begin{pmatrix} 0 & LR & LR \\ RL & 0 & TB \\ RL & BT & 0 \end{pmatrix}, \quad S'_2 = \begin{pmatrix} 0 & TB & TB \\ BT & 0 & LR \\ BT & LR & 0 \end{pmatrix}.$$

For example, the characters $\begin{smallmatrix} \text{𠂇} \\ \text{𠂇} \\ \text{𠂇} \end{smallmatrix}$ and $\begin{smallmatrix} \text{𠂇} \\ \text{𠂇} \\ \text{𠂇} \end{smallmatrix}$ belong to section s_1 , and $\begin{smallmatrix} \text{𠂇} \\ \text{𠂇} \\ \text{𠂇} \end{smallmatrix}$ and $\begin{smallmatrix} \text{𠂇} \\ \text{𠂇} \\ \text{𠂇} \end{smallmatrix}$ belong to section s_2 .

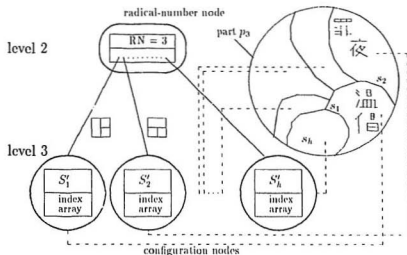


Figure 6.2. Tree representing the partition of a part into sections.

A *group* is represented by a combination node at the fourth level of the tree. The *SNC* of the group is stored in the node as its key. The combination node with $SNC = [q_1, \dots, q_n]$ is pointed to by the index array element $A[q_1, \dots, q_n]$ of its parent configuration node (see Figure 6.3). For example, the characters in the section s_2 have two radicals. The index array A of the configuration node for the section is a 2-dimensional array. The section is split into several groups: $\dots, g_{k,j}, \dots$ where the *SNC* of $g_{k,j}$ is $[k, j]$. The group $g_{k,j}$ is represented by a combination node which is pointed to by the index array element $A[k, j]$. For characters $\begin{array}{c} \times \\ | \end{array}$ and $\begin{array}{c} \uparrow \\ \times \end{array}$ which have two radicals, their *SNC*s are $[2, 3]$. Both characters belong to group $g_{2,3}$ represented by a combination node pointed to by the index array element $A[2, 3]$ (see Figure 6.3).

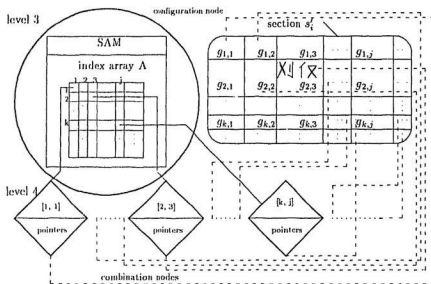


Figure 6.3. Tree representing the partition of a section into groups.

The group with the same SNC can be subdivided into "same 1-radical groups" (SIGs). For example, in Figure 6.4, group $g_{3,4}$ which belongs to the subtree with $RN = 2$, is divided into $\sum -SIG$, 1^+ -SIG, etc.

A character node in the tree is represented by a leaf node and corresponds to a Chinese character. For a subtree with $RN = 1$, the leaves are situated at the 5th level of the tree; in a subtree with $RN = 2$, the leaves are at the 6th level; for a subtree with $RN = k$, the leaves are located at the $(k + 4)$ th level. A leaf node contains the RAM of the last radical of the character, and a radical node at the 5th level contains the RAM of the first radical. Generally, the radical node at the $(k + 4)$ th

level contains the *RAM* of the k th radical.

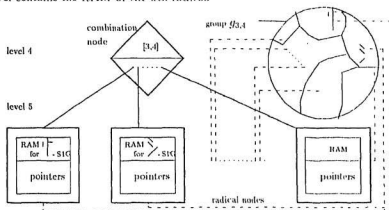


Figure 6.4. Tree representing the partition of a group.

As an example, Figure 6.5 shows how character $\frac{\text{ㄗ}}{\text{X}}$ is represented by a path containing the radical nodes and the character node in the tree. The first radical of the character has three strokes, the second has four strokes, and the third has three strokes. The path starts from the combination node with $RNC = [3,4,3]$ at the 4th level of the tree. The radical node on the 5th level of the tree represents the first radical ㄗ by the *RAM* of the radical; the radical node at the 6th level represents the second radical ㄗ ; the character node at the 7th level represents the third radical X .

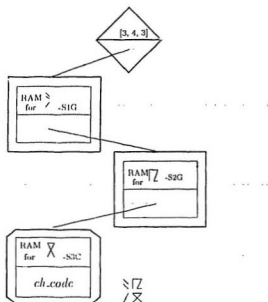


Figure 6.5. Representing character $\sim \square / X$ with a path in the tree.

The overall heterogeneous multi-way tree organization of the database is shown in Figure 6.6.

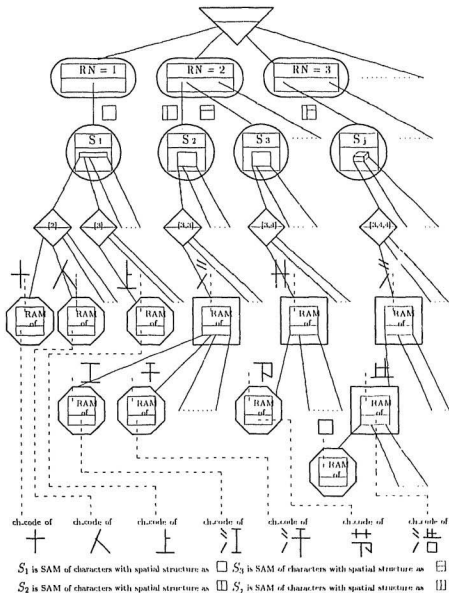


Figure 6.6 Heterogeneous multi-way tree organization of model database.

6.3 Construction of multi-way tree

Before searching the model database to recognize an input handwritten Chinese character, the tree must first be constructed. The proposed insertion algorithm for constructing the multi-way tree is:

1. Create the root representing the model database.
2. Insert each model(HAGR) into the tree according to the following steps:
 - (1) Derive the radical number($RN: n$), the structure adjacency matrix($SAM: S$), and the stroke number combination($SNC: [q_1, \dots, q_n]$) from the model(HAGR).
 - (2) Start from the root of the tree, search at the second level. If the node with $RN = n$ does not exist, a new child(a radical-number node) with $RN = n$ is inserted.
 - (3) From the node with $RN = n$, search at the third level. If the node with $SAM = S$ does not exist, a new child(a configuration node) with $SAM = S$ is inserted.
 - (4) Access index array element $A[q_1, \dots, q_n]$ of the node with $SAM = S$. If the element is *nil*, a new node with $SNC = [q_1, \dots, q_n]$ at the fourth level is created.
 - (5) From the node with $SNC = [q_1, \dots, q_n]$, search its children (at the fifth level) with the first radical adjacency matrix($RAM = R_1$) of the model. When the child with $RAM = R_1$ cannot be found, if the model has more than one radical, a new child(radical node) with $RAM = R_1$ is inserted, and proceed to the next

step. If the model has only one radical, a character node with $RAM = R_1$ is inserted.

- (6) At the $(j + 4)$ th level ($j \geq 1$), for the radical node with $RAM = R_j$, search its children at the $(j + 5)$ th level with the $(j + 1)$ th radical adjacency matrix ($RAM = R_{j+1}$) of the model. If a child with $RAM = R_{j+1}$ cannot be found and the model has radicals more than $j + 1$, a new child(radical node) with $RAM = R_{j+1}$ is inserted, increase j , and repeat the step. If the model has $j + 1$ radicals, the character node with $RAM = R_{j+1}$ for the model is inserted.

The following procedure expressed in pseudo-codes performs the construction of the tree.

Procedure {Tree Construction }

begin

 Create the root of the tree

for each model $H_m = [X_m, E_m]$ **do**

begin { insert models }

 derive $RN: n, SAM: S, SNC: [q_1, \dots, q_n]$ from H_m

 search children(radical-number node) of the root

if the node with $RN = n$ **is not found then**

 insert a radical-number node(new child) with $RN = n$

 from the radical-number node with $RN = n$, search its children (configuration nodes) with S

if the node with $SAM = S$ **does not exist then**

 insert a configuration node(new child) with $SAM = S$

 from the configuration node with $SAM = S$, access its index array element $A[q_1, \dots, q_n]$

if $A[q_1, \dots, q_n]$ **is empty then**

 create a new combination node $[q_1, \dots, q_n]$ pointed to by $A[q_1, \dots, q_n]$

 from the combination node $[q_1, \dots, q_n]$, search its children with R_1

```

if the node with  $RAM = R_1$  is not found then
  if  $(n > 1)$  {more than one radical exist} then
    create a radical node(new child) with  $RAM = R_1$ 
  else {there is only one radical} do
    create a character node(new child) with  $RAM = R_1$ 
  for  $j = 2$  to  $n$  do
    begin
      from the radical node with  $RAM = R_{j-1}$ , search its children with  $R_j$ 
      if the node with  $RAM = R_j$  is not found then
        if  $(n > j)$  {there are more than  $j$  radicals} then
          create a radical node(new child) with  $RAM = R_j$ 
        else {there are only  $j$  radicals} do
          create a character node(new child) with  $RAM = R_j$ 
        end
      end
    end
  end
end
end

```

6.4 Searching the multi-way tree for character recognition

With the heterogeneous multi-way tree, a global decision can be made via a series of simple and local decisions at the different levels of the tree to ascertain the presence or absence of the model of an input character in the database. In this systematic manner, the efficiency and accuracy of the matching process for recognizing the input character can be greatly improved.

6.4.1 Searching algorithm

Once the IIAGR of a character has been constructed, the radical number($RN : n$), structure adjacency matrix($SAM : S$), and stroke number combination ($SNc' : [q_1, \dots, q_n]$) of the character can be derived.

The search starts from the root. If one of its children(radical-number nodes) with $RN = n$ is found at the second level of the tree, its children(configuration nodes) at the third level are searched. If S is matched by the SAM of a configuration node, the index array element $A[q_1, \dots, q_n]$ of the matched node is accessed. The children of the combination node pointed to by $A[q_1, \dots, q_n]$ are visited to see if the RAM of a child matches with the first RAM of the input character. If the matched node at the fifth level of the tree is a leaf(character node) and the character has only one radical, the input character is recognized and its character code is equal to the ch_code of the node. Otherwise, the search process continues with the successors of the node until the RAM of a leaf is matched by the RAM of the last radical of the input character. The input character is recognized when a path from the root of the tree to a leaf is found such that all the nodes along the path are matched. Otherwise, the input character should be rejected.

This algorithm is presented in pseudo-code as follows:

Input : $II_i = [X, E]$

$R_i (i = 1, \dots, n)$

Output : Character code or message to reject the character

Procedure {Search tree to recognize the input character}

begin

 derive $RN : n$, $SAM : S$, and $SNC : [q_1, \dots, q_n]$ from H_i ;

 search the children(radical-number nodes) of the root

if there is no such node with $RN = n$ **then**

 Reject H_i ; exit;

 from the radical-number node with $RN = n$, search its children

if there is no such node with SAM matched by S **then**

 Reject H_i ; exit;

 from the configuration node with $SAM = S$, access its index array element

$A[q_1, \dots, q_n]$.

if the element is empty **then**

 Reject H_i ; exit;

 from the combination node $[q_1, \dots, q_n]$, search its children

if no node with RAM matched by R_1 **then**

 Reject H_i ; exit;

$j = 1$;

while the node matched by R_j is not a leaf node **do**

begin

 search the children of the matched radical node

if no child with RAM matched by R_{j+1} **then**

 Reject H_i ; exit;

else

$j = j + 1$.

end

 Print the character code of the input character

end

For example, input character 扌 is composed of three radicals 扌, 日, and

$\overline{\Gamma}$. Its character adjacency matrix (CAM) is:

$$B = \begin{pmatrix} 1 & LR & LR \\ RL & 5 & TB \\ RL & BT & 3 \end{pmatrix}.$$

The RAMs of the three radicals are:

$$R_1 = \begin{pmatrix} 0000101 & 0100000 & 0000000 & 0000000 \\ 0100000 & 0010001 & 0100000 & 0000001 \\ 0000000 & 0100000 & 0001000 & 0000000 \\ 0000000 & 0000001 & 0000000 & 0100010 \end{pmatrix},$$

$$R_2 = \begin{pmatrix} 0000101 & 0000001 & 0000000 & 0000001 & 0000000 \\ 0000001 & 0010001 & 0000010 & 0000000 & 0000001 \\ 0000000 & 0000010 & 0000101 & 0000100 & 0000000 \\ 0000001 & 0000000 & 0000100 & 0010001 & 0000001 \\ 0000000 & 0000001 & 0000000 & 0000001 & 0000101 \end{pmatrix},$$

$$R_3 = \begin{pmatrix} 0000101 & 0000000 & 0001000 \\ 0000000 & 0000101 & 0100000 \\ 0001000 & 0100000 & 0010001 \end{pmatrix}.$$

From matrix $B(CAM)$, the RN , SAM , and SNC of the character can be derived

as illustrated below:

$$HN: 3$$

$$SAM: S = \begin{pmatrix} 0 & LR & LR \\ RL & 0 & BT \\ RL & BT & 0 \end{pmatrix}$$

$$SNC: [4, 5, 3].$$

Radical-number node with $RN = 3$ is found at the second level of the tree (Figure 6.7). A configuration node with its structure adjacency matrix (SAM) matched by S is found at the third level. With SNC of the input character, the combination node pointed to by the index array element $A[4, 5, 3]$ is reached. RAM of a successor of the combination node matches that of the first radical $𠂇$ of the input character. The search continues to match the second radical $𠂇$ and then the third radical $𠂇$ of the input character. As the last matching node is a leaf (character node), the model of the input character has been found and the input character is therefore recognized.

The recognition of an input character by searching the model of the character in the database is quite fast and accurate because the models with different $SAMs$, $SNCs$, or unmatched $RAMs$, are excluded from the searching at the different levels. In the searching process, at any level of the tree, if no node can be matched before a matched leaf is found, the input character is rejected and the search is terminated.

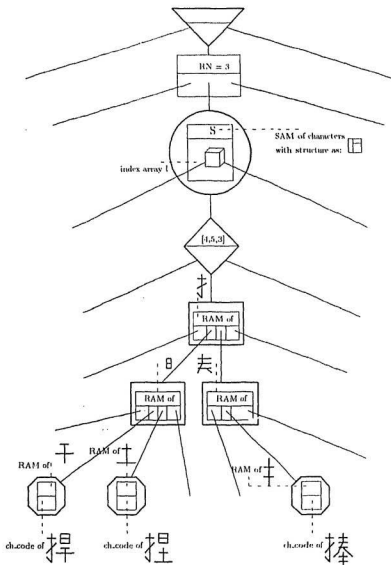


Figure 6.7. Heterogeneous multi-way tree of character models.

6.4.2 Time complexity analysis of searching algorithms

Suppose an input character has n radicals ($RN = n$). Its structure adjacency matrix(SAM) is $S_{n \times n}$, and its stroke number combination(SNC') is $[q_1, \dots, q_n]$. The RAM s for its 1st, \dots , n th radicals are $(R_1)_{q_1 \times q_1}, \dots, (R_n)_{q_n \times q_n}$. Let radical number n , the entries of $S_{n \times n}$, q_i ($0 < i \leq n$), and the entries of $(R_i)_{m_i \times m_i}$ ($0 < i \leq n$) be features.

As matrix $S_{n \times n}$ is a symmetric matrix and only the entries in its triangle section need to be used, the total number of entries is $\sum_{i=1}^n i$. Similarly, matrix $(R_i)_{q_i \times q_i}$ is a symmetric matrix and the total number of its entries is $\sum_{j=1}^{q_i} j$. Therefore, the total number of features N_f of the input character which can be used to search the model database would be:

$$N_f = 1 + \sum_{i=1}^n i + n + \sum_{i=1}^n \sum_{j=1}^{q_i} j = (n+2)(n+1)/2 + \sum_{i=1}^n (q_i+1)q_i/2. \quad (6.1)$$

An important criterion to judge an algorithm is the amount of time needed to find the model of an input character. However, the time to reject an input character is another important criterion since quite often the input character is written too carelessly by writers. In this situation, it is desirable that the character be rejected as soon as possible.

6.4.2.1 Criteria for rejection of an input character

Assuming that there are M models in the model database, for the sequential linear list searching algorithm, the best case is that only one feature is used to exclude all models in the list, thus requiring M comparisons. Using the multi-step linear list

searching algorithm, the best case is that in the first step all models are excluded by only one feature and the number of comparison is M .

We can show that the tree searching algorithm in an average case is better than the previous two algorithms in the best case. Let c_i be the average number of children of a node at the i level of tree. On the average, the search time to move from the first level to the second level is $c_1/2$; the time to move from the second level to the third level is $((c_2/2) \sum_{i=1}^n i)/2$. The search from the third level to the fourth goes through an index array and no comparison is needed. Similarly, the time to search from the $(i+3)$ th level to the $(i+4)$ th level is $((c_{i+3}/2) \sum_{j=1}^{m_i} j)/2$ ($1 \leq i \leq n$). Thus, the time to reject an input character is:

$$T_{tr} = c_1/2 + ((c_2/2) \sum_{i=1}^n i)/2 + \sum_{i=1}^n ((c_{i+3}/2) \sum_{j=1}^{q_i} j)/2 \quad (6.2)$$

Our database contains 3300 models of daily used characters. According to the statistics, $c_1 = 8$, $c_2 = 8$, $c_3 = 16$, $c_4 = c_5 = \dots = 6$, and the average number of radicals in an input character is $n = 3$. Most radicals have no more than ten strokes ($m_i < 10$). In an average case, the time to reject an input character is $4 + 12 + 83 + 83 + 83 = 265 \ll 3300$.

6.4.2.2 Criteria for recognition of an input character

With the sequential linear list searching algorithm, on the average, for each feature of an input character, only half the number($M/2$) of models in the list need to be compared. Therefore, the average number of comparisons is:

$$T_{sf} = N_f(M/2) = (n+1)(n+2)M/4 + M \sum_{i=1}^n (q_i + 1)q_i/4. \quad (6.3)$$

For the multi-step linear list searching algorithm, assume that the search is divided into $n+3$ steps. RN (one features), $S_{n \times n}(n(n+1)/2 \text{ features})$, $[q_1, \dots, q_n](n \text{ features})$, $R_1(m_1(m_1+1)/2 \text{ features})$, \dots $R_n(m_n(m_n+1)/2 \text{ features})$ are used correspondingly in the first step, \dots , $n+3$ th step. N_i is the number of models remaining in the list at the i th step. On the average, only half the number of models that remain in the list should be compared in each step. Hence, the average number of comparisons is:

$$T_{mf} = M/2 + (N_2/2)n(n+1)/2 + nN_3/2 + \sum_{i=1}^n (N_{i+3}/2)q_i((q_i+1)/2). \quad (6.4)$$

Comparing with the sequential linear list searching algorithm, the average time reduced by the multi-step searching algorithm list is given by:

$$T_{sf} - T_{mf} = n(n+1)(M - N_2)/4 + n(M - N_3)/2 + \sum_{i=1}^n q_i(q_i+1)(M - N_{i+3})/4. \quad (6.5)$$

For the tree searching algorithm, its RN (one features), $S_{n \times n}(n(n+1)/2 \text{ features})$, $[q_1, \dots, q_n](n \text{ features})$, $R_1(m_1(m_1+1)/2 \text{ features})$, \dots $R_n(m_n(m_n+1)/2 \text{ features})$ are used to search the nodes correspondingly at the second level, \dots , $n+4$ level. To find the model of an input character, a path from the root to the character node has to be found. Each node along the path is matched by the corresponding features of the input character. Suppose that the matched node at i th level on the path has f_i children. Again, on the average, only half the number of children of each matched node should be compared to find the path except for matched configuration node. The average number of comparisons in this case is:

$$T_{if} = f_1/2 + (f_2/2)n(n+1)/2 + \sum_{i=1}^n (f_{i+3}/2) \sum_{j=1}^{q_i} j. \quad (6.6)$$

The average time reduced by comparing the tree searching algorithm to the multi-step searching algorithm is given by:

$$T_{mf} - T_{lf} = (N_1 - f_1)/2 + (N_2 - f_2)n(n+1)/4 + N_3n/2 + \sum_{i=1}^n (N_{i+3} - f_{i+3})q_i(q_i+1)/4. \quad (6.7)$$

Here, N_i is the number of models which remained in the list at the i th step, f_i is the number of children of the matched node on i th level of the tree. Hence, $N_i \gg f_i$ and $T_{mf} \gg T_{lf}$.

6.5 Memory reduction

With the tree organization of the models, searching the database for the recognition of a character is divided into a number of simple and local decisions at different levels of the tree. However, quite a large memory storage is required for the tree organization. In the actual implementation, the memory required for the heterogeneous multi-way tree is greatly reduced by introducing a radical linked list and compression vectors.

6.5.1 Memory reduction with radical linked list

According to our analysis, Chinese characters are composed of 200 basic radicals. Different characters may share the same radicals. A radical contained in several characters would be duplicated several times for tree organization. The duplication problem is very serious in the tree organization method reported in [Chen, et al.

1988]. In that method, a character is represented by a path from the root to a leaf of the tree. If two characters share a radical, the radical would be duplicated in the two paths corresponding to the characters. In other words, a segment of one path represents a radical which would also appear in the path of another character.

The radical duplication problem in the heterogeneous multi-way tree organization is overcome by introducing the radical linked list. Instead of duplicating the same *RAM* for the common radical at different nodes in the tree, only one copy of the *RAM* is stored in a linked list. The address of the *RAM* in the radical list is recoded as the key in those radical nodes. The linked list called *radical list* (see Figure 6.8) stores the radical adjacency matrices of all radicals. A node in the radical list has three fields: the radical code, the radical adjacency matrix, and the pointer.

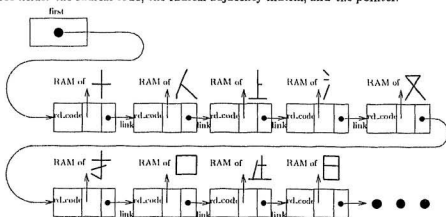


Figure 6.8. Radical list of *RAMS*.

For example, characters 彳 and 浩 share the same radical 彳 in different subtrees (see Figure 6.6). Only one copy of the *RAM* of the radical is required in the

radical list, with the address of the *RAM* being recoded in the corresponding radical nodes for these characters.

Assume that M_b is the total number of radicals composing the character models in the database; d_i is the number of duplications for radical i . Assume as well that one unit of storage is required for an address or a matrix entry. The number of storage units needed to store the radical adjacency matrix(*RAM*) of the radical is u_i , and the amount of memory required for storing the radical adjacency matrices without the radical list is $\sum_{i=1}^{M_b} d_i \cdot u_i$. With the radical list, the amount of memory required is:

$$\sum_{i=1}^{M_b} d_i \cdot 1 + \sum_{i=1}^{M_b} u_i = \sum_{i=1}^{M_b} (d_i + u_i). \quad (6.8)$$

The memory reduction is: $\sum_{i=1}^{M_b} (d_i \cdot u_i - d_i - u_i)$. According to our statistics, the average memory u_i for matrices is about 25 units, and the average number of duplications for a radical (d_i) is around 10. The memory saving is very significant, since $d_i \cdot u_i = 250$ is much bigger than $d_i + u_i = 35$. During searching, the key of a radical node is required for *RAM* matching. The address used by the *RAM* to access the radical adjacency matrix in the radical linked list. The searching time of the multi-way tree is only increased by the indirect addressing (dereferencing) time.

6.5.2 Memory reduction with compression vectors

Let m_i be the number of configuration nodes representing the characters with i radicals. Suppose that the maximum number of strokes in a radical is n . The total

number of elements for these i -dimension index arrays in the configuration nodes with i radicals is $m_i u^i$. The total number of elements for the index arrays in the multi-way tree is $\sum_{i=1}^N m_i u^i$, where N is the maximum number of radicals for Chinese characters.

Most Chinese characters (about 98% characters in XinHua Dictionary) have no more than three radicals. The total number of Chinese character with more than three radicals is quite small (less than 1000). All the index arrays with more than three dimensions ($i > 3$) would have $\sum_{i=4}^N m_i u^i$ elements. By the pigeonhole principle, at least $(\sum_{i=4}^N m_i u^i - 1000)$ elements are empty and wasted. Therefore, index arrays for the characters with more than three radicals are quite sparse.

According to statistical data, most radicals have no more than ten strokes. For this reason, even for the two-dimensional index array A_2 , the part of the array containing the elements $A_2[i, j]$ ($i > 10$ or $j > 10$) is very sparse. Similarly, the part of a three-dimensional index array A_3 consisting of the elements $A_3[i, j, k]$ ($i > 10$, $j > 10$ or $k > 10$) is also very sparse.

Two techniques are used to overcome the potential waste of memory:

1. For the configuration nodes representing the characters with two (or three) radicals, the elements (pointers) are organized by two different structures.
 - A two (or three) dimensional index array organizes the pointers of a configuration node representing characters with two (or three) radicals. Each radical has no more than ten strokes.

- A compression vector organizes the pointers of a configuration node of two (or three) radicals, one of which has more than ten strokes.
2. For the configuration node representing the characters with more than three radicals, its pointers are organized by a compression vector instead of an index array.

A compression vector is a one-dimensional array (see Figure 6.9). For each element of the array, its first field code encodes the SNC' $\{q_1, \dots, q_n\}$ by formula

$$code = q_1 \times b^{n-1} + \dots + q_{n-1} \times b_1 + q_n. \quad (6.9)$$

where $b >$ maximum number of radicals in a character. The second attribute pointer points to a combination node.

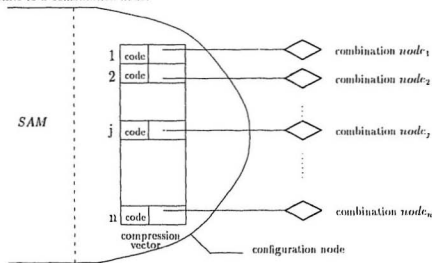


Figure 6.9. Compression vector.

The elements of a compression vector are ordered in a non-decreasing sequence according to the codes of the elements. The purpose is to use the binary search to select the combination node with *SNC* for an input character. Most characters organized by index arrays can be visited by directly accessing the index array elements. The small portion of characters organized by a compression vector can be visited by binary search. Therefore, searching for character recognition could still be very fast, but the memory reduction is very significant.

Chapter 7

Learning by Samples for Model Development

7.1 Introduction

Hierarchical attributed graph representation (HAGR) can describe and represent both invariant and unstable features of a Chinese character written in different styles. The variations related to the stroke orientation, stroke type, and relation between strokes can be represented in a hierarchical attributed graph (HAG) without increasing the number of models for each Chinese character in the model database. Learning by samples is introduced to derive a model from its samples written by different people with various handwriting styles and habits. Furthermore, graph synthesis is introduced to build models for new characters and to update the models in a database to

represent more handwriting variations of corresponding Chinese characters.

Suppose that the HAG constructed from a sample is called SHAG. Likewise, the radical attributed graph (RAG) for a component radical of the sample is called SRAG; the HAG of a model is called MHAG and the RAG of a component radical in the model is called MRAG. During the learning phase, several SHAGs of a new character are derived and then synthesized. For a character with its model in the database, if the models of a character exist in the database, then the models can be updated by synthesizing the MHAGs of these models with the SHAGs of the character. Other approaches [Cheng et. al 1989; Nagy 1988; Saito 1985; Gu 1983] require, for a single character, many models for its samples. Furthermore, it is difficult to determine how many samples are required to obtain a complete description for a character in the database. Using a graph synthesis process, only a few MHAGs need to be built for a character.

7.2 Attributed graph synthesis

During the synthesis process, synthesis evaluation is applied to determine whether two original graphs can be synthesized. While original graphs are compared, each comparison is systematically described by a skeleton graph for the purpose of synthesis evaluation. If there is a skeleton graph which satisfies a set of given conditions, the two original graphs will be synthesized into a convergent graph. There are two kinds of graphs synthesis: RAG synthesis which is used to integrate two RAGs, and HAG

synthesis which is used to integrate two HAGs.

Definition 7.1 A *skeleton graph* $G_s = (V_s, E_s)$ is an attributed graph associated with two original graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with the same number of vertices such that each vertex (C-NODE) $v_{ij} \in V_i$ describes the comparison of two vertices $v'_i \in V_1$ and $v''_j \in V_2$ and each edge (C-LINK) $e_{ij} \in E_s$ describes the comparison of two edges $e'_{ij} \in E_1$ and $e''_{kl} \in E_2$. In other words, the skeleton graph contains the information which is obtained by comparisons of corresponding edges and vertices.

Definition 7.2 *Type* is an attribute for an edge (C-LINK) or a vertex (C-NODEs) of a skeleton graph where its value represents the comparison results of two corresponding edges or vertices in two original graphs.

7.2.1 Synthesis evaluation of two RAGs

Suppose that two radical attributed graphs are $R_1 = (V_1, E_1)$ and $R_2 = (V_2, E_2)$ where $V_1 = \{v'_1, v'_2, \dots, v'_n\}$, $E_1 = \{\dots, e_{pq}, \dots\}$, $V_2 = \{v''_1, v''_2, \dots, v''_n\}$, and $E_2 = \{\dots, e''_{st}, \dots\}$. Let $A = [a_{ij}]_{n \times n}$ and $B = [b_{kl}]_{n \times n}$ be two radical adjacency matrices (RAMs) corresponding to R_1 and R_2 , respectively. As well, let the corresponding skeleton graph be $R_s = (V_s, E_s)$.

The comparison results of two vertices in two RAGs could be exactly matched (e-match), partially matched (p-match), nearly matched (n-match), or unmatched (u-match).

An *e-match* indicates that two vertices v'_i and v''_k exactly match with each other

such that $a_{ii}(r) = b_{kk}(r)$, ($\forall r$, $0 \leq r \leq 5$), where l means the l th right most bit. The stroke described by v'_i is exactly the same as that described by v''_k . The logical function determining whether both vertices exactly match with each other is:

$$\begin{aligned} F_e = & [(a_{ii}(0) \cdot (b_{kk}(0)) \vee (\overline{a_{ii}(0)} \cdot \overline{b_{kk}(0)}))] \cdot [(a_{ii}(1) \cdot (b_{kk}(1)) \vee (\overline{a_{ii}(1)} \cdot \overline{b_{kk}(1)}))] \cdot \\ & [(a_{ii}(2) \cdot (b_{kk}(2)) \vee (\overline{a_{ii}(2)} \cdot \overline{b_{kk}(2)}))] \cdot [(a_{ii}(3) \cdot (b_{kk}(3)) \vee (\overline{a_{ii}(3)} \cdot \overline{b_{kk}(3)}))] \cdot \\ & [(a_{ii}(4) \cdot (b_{kk}(4)) \vee (\overline{a_{ii}(4)} \cdot \overline{b_{kk}(4)}))] \cdot [(a_{ii}(5) \cdot (b_{kk}(5)) \vee (\overline{a_{ii}(5)} \cdot \overline{b_{kk}(5)}))] \quad (7.1) \end{aligned}$$

A *p-match* indicates that two vertices v'_i and v''_k partially match with each other, such that $\exists h$, $2 \leq h \leq 5$, $a_{ii}(h) = b_{kk}(h) = 1$. For example, v'_i describes a line stroke which can be written as a horizontal or right diagonal ($a_{ii} = 001101$), and v''_k describes a line stroke which can be written as a right diagonal(RD) ($b_{kk} = 001001$). Since $a_{ii}(3) = b_{kk}(3) = 1$, v'_i and v''_k are partially matched. If two vertices partially match with each other, one variation of direction described by the first vertex is also described by another vertex. In this case, it is thought that the two vertices describe the variations of the same stroke. The logical function determining whether both vertices partially match is:

$$F_p = [a_{ii}(2) \cdot b_{kk}(2)] \vee [a_{ii}(3) \cdot b_{kk}(3)] \vee [a_{ii}(4) \cdot b_{kk}(4)] \vee [a_{ii}(5) \cdot b_{kk}(5)] \quad (7.2)$$

An *n-match* indicates that the two vertices v'_i and v''_k nearly match such that $\exists h, l$, $2 \leq h, l \leq 5$, $a_{ii}(l) = b_{kk}(h) = 1$ and $|l - h| = 1$ or 3 . For example, v'_i describes a horizontal(H) line stroke and v''_k describes a right diagonal(RD) stroke, such that $a_{ii} = 000101$ and $b_{kk} = 001001$. Since $a_{ii}(2) = b_{kk}(3)$ and $|2 - 3| = 1$, v'_i is

nearly matched with v_k'' . When two vertices nearly match, one variation of direction described by the first vertex is 45° different from one of the variations of direction described by the other. Because the direction of a handwritten stroke can be changed by no more than 45° , it can be thought that the two vertices describe the variations of the same stroke. The logical function determining whether both vertices nearly match is:

$$\begin{aligned}
 F_n = & [a_{ii}(2) \cdot b_{kk}(2)] \vee [a_{ii}(3) \cdot b_{kk}(3)] \vee [a_{ii}(4) \cdot b_{kk}(4)] \vee [a_{ii}(5) \cdot b_{kk}(5)] \vee \\
 & [a_{ii}(2) \cdot b_{kk}(3)] \vee [a_{ii}(3) \cdot b_{kk}(2)] \vee [a_{ii}(3) \cdot b_{kk}(4)] \vee [a_{ii}(4) \cdot b_{kk}(3)] \vee \\
 & [a_{ii}(2) \cdot b_{kk}(5)] \vee [a_{ii}(5) \cdot b_{kk}(2)] \vee [a_{ii}(4) \cdot b_{kk}(5)] \vee [a_{ii}(5) \cdot b_{kk}(4)] \quad (7.3)
 \end{aligned}$$

An *u-match* indicates that two vertices do not match at all. The strokes described by them are different. The logic function determining whether both vertices are *u-match* is:

$$F_u = \overline{F_v \vee F_p \vee F_n} \quad (7.4)$$

According to the analyses based on the logical functions given by (7.1), (7.2), (7.3), and (7.4), a logical operator \odot_u between a_{ii} and b_{kk} can be defined to determine whether two vertices v_i' and v_k'' describe the same stroke.

$$\begin{aligned}
 a_{ii} \odot_u b_{kk} &= F_v \vee F_p \vee F_n \vee F_u \\
 &= [a_{ii}(2) \cdot (b_{kk}(2) \vee b_{kk}(3) \vee b_{kk}(5))] \vee \\
 &\quad [a_{ii}(3) \cdot (b_{kk}(2) \vee b_{kk}(3) \vee b_{kk}(4))] \vee
 \end{aligned}$$

$$\begin{aligned}
& [a_{ii}(4) \cdot (b_{kk}(3) \vee b_{kk}(4) \vee b_{kk}(5))] \vee \\
& (a_{ii}(5) \cdot (b_{kk}(2) \vee b_{kk}(4) \vee b_{kk}(5))) \quad (7.5)
\end{aligned}$$






It can be easily proved that $a_{ii} \odot_v b_{kk} = b_{kk} \odot_v a_{ii}$. For two given vertices v_i^j and v_k^u the type value of corresponding C-NODE v_i^j ($v_i^j \in V_s$) is defined such that:

$$TYPE(v_i^j) = \begin{cases} 1, & \text{if } v_i \text{ and } v_k \text{ are c-match, p-match, or n-match} \\ 0, & \text{if } v_i \text{ and } v_k \text{ are u-match} \end{cases} \quad (7.6)$$

The comparison results of two edges in two RAGs could be exact-match(e-match), loosely match(l-match), approximately match(a-match), or unmatched(u-match).






An *e-match* indicates that two edges e_{ij}^j and e_{st}^u are matched as a relation described by the first edge is the same as a relation described by the second edge. The logical function to determine whether two edges are e-match is:

$$\begin{aligned}
B_e = & [(a_{ij}(0) \cdot b_{st}(0)) \vee (\overline{a_{ij}(0)} \cdot \overline{b_{st}(0)})] \cdot [(a_{ij}(1) \cdot b_{st}(1)) \vee (\overline{a_{ij}(1)} \cdot \overline{b_{st}(1)})] \cdot \\
& [(a_{ij}(2) \cdot b_{st}(2)) \vee (\overline{a_{ij}(2)} \cdot \overline{b_{st}(2)})] \cdot [(a_{ij}(3) \cdot b_{st}(3)) \vee (\overline{a_{ij}(3)} \cdot \overline{b_{st}(3)})] \cdot \\
& [(a_{ij}(4) \cdot b_{st}(4)) \vee (\overline{a_{ij}(4)} \cdot \overline{b_{st}(4)})] \cdot [(a_{ij}(5) \cdot b_{st}(5)) \vee (\overline{a_{ij}(5)} \cdot \overline{b_{st}(5)})] \quad (7.7)
\end{aligned}$$

An *l-match* indicates that edge e_{ij}^j represents a loose relation and edge e_{st}^u does not exist. Relations \top , \perp , \vdash , \neg , and L are loose relations. This means that if the relation between two strokes is a loose relation, these two strokes could disconnect with each other. For example, radical  can be written as  or . The right vertical stroke and the bottom horizontal stroke have relation L in  but do not connect with each other in . The logical function to determine whether two

edges are l-match is:

$$\begin{aligned}
 B_l = & \{[(a_{ij}(0) \cdot \overline{b_{st}(0)}) \vee (\overline{a_{ij}(0)} \cdot b_{st}(0))] \cdot [(\overline{a_{ij}(1)} \cdot \overline{b_{st}(1)}) \vee (\overline{a_{ij}(1)} \cdot b_{st}(1))]\} \cdot \\
 & [(a_{ij}(2) \cdot \overline{b_{st}(2)}) \vee (\overline{a_{ij}(2)} \cdot b_{st}(2))] \cdot [(\overline{a_{ij}(3)} \cdot \overline{b_{st}(3)}) \vee (\overline{a_{ij}(3)} \cdot b_{st}(3))]\} \cdot \\
 & [(a_{ij}(4) \cdot \overline{b_{st}(4)}) \vee (\overline{a_{ij}(4)} \cdot b_{st}(4))]\} \cdot [(a_{ij}(5) \vee \overline{b_{st}(5)}) \cdot (\overline{a_{ij}(5)} \vee b_{st}(5))]\} \quad (7.8)
 \end{aligned}$$

An *a-match* indicates that the pair of relations represented by e'_{ij} and e''_{st} is a pair of quasi-matching relations. The pairs of quasi-match relations are (L, \top) , (L, \perp) , (L, \vdash) , and (L, \dashv) . If the relation between two strokes belong to a pair of quasi-match relations, these two strokes could have either an *L* connection or any one of the connections \top , \perp , \vdash , and \dashv . For example, radical  can be written as  or . The left vertical stroke and the bottom horizontal stroke have relation *L* in  but has relation \vdash in . The logical function to determine whether two edges are an *a-match* is:

$$\begin{aligned}
 B_o = & [a_{ij}(0) \cdot (b_{st}(1) \vee b_{st}(2) \vee b_{st}(3) \vee b_{st}(4))] \vee \\
 & [b_{st}(0) \cdot (a_{ij}(1) \vee a_{ij}(2) \vee a_{ij}(3) \vee a_{ij}(4))] \quad (7.9)
 \end{aligned}$$

An *u-match* indicates that two edges are not matched at all. The logical function to determine whether two edges are u-match is:

$$B_u = \overline{B_e \vee B_l \vee B_a} \quad (7.10)$$

Summarizing (7.7), (7.8), (7.9), and (7.10), whether two vertices e'_{ij} and e''_{st} are matched can be determined by a logical operator \odot_e between a_{ij} and b_{st} such that:

$$a_{ij} \odot_e b_{st} = B_e \vee B_l \vee B_a \vee B_u$$

$$\begin{aligned}
&= [\overline{(a_{ij}(5) \cdot b_{st}(5)) \vee (a_{ij}(5) \cdot b_{st}(5))}] \cdot \\
&\quad [\overline{a_{ij}(1)} \vee (a_{ij}(1) \cdot \overline{b_{st}(2) \vee b_{st}(3) \vee b_{st}(4)})] \cdot \\
&\quad [\overline{a_{ij}(2)} \vee (a_{ij}(2) \cdot \overline{b_{st}(1) \vee b_{st}(3) \vee b_{st}(4)})] \cdot \\
&\quad [\overline{a_{ij}(3)} \vee (a_{ij}(3) \cdot \overline{b_{st}(1) \vee b_{st}(2) \vee b_{st}(4)})] \cdot \\
&\quad [\overline{a_{ij}(4)} \vee (a_{ij}(4) \cdot \overline{b_{st}(1) \vee b_{st}(3) \vee b_{st}(2)})] \quad (7.11)
\end{aligned}$$

It can be easily shown that $a_{ij} \odot_r b_{st} = b_{st} \odot a_{ij}$. For two given edges e'_{ij} and e''_{st} , the type value of corresponding C-LINK e^c_{ij} ($e^c_{ij} \in E_s$) is such that:

$$TYPE(e^c_{ij}) = \begin{cases} 1, & \text{if } e'_{ij} \text{ and } e''_{st} \text{ are e-match, l-match, or a-match} \\ 0, & \text{if } e'_{ij} \text{ and } e''_{st} \text{ are u-match} \end{cases} \quad (7.12)$$

The skeleton graph can be reduced by removing those C-NODES and C-LINKS with type value equal to 0. The reduction of the skeleton graph for two radical graphs R_1 and R_2 includes the following steps:



1. Remove the C-LINKS and C-NODES with type value equal to 0.
2. Delete the C-NODES connected to the removed C-LINKS.
3. Delete the C-LINKS incident to the removed C-NODES.

Definition 7.3 A reduced skeleton graph R_r for two radical graphs R_1 and R_2 with the same number of vertices is called completion of R_1 and R_2 if $order(R_r) = order(R_1)$. Two radical graphs can be synthesized if their completion exists. A reduced skeleton graph R_r can be represented by an incidence matrix T with diagonal entries describing

the C-NODEs and nondiagonal entries describing the C-LINKs. The incidence matrix can be written as:

$$T = [t_{pq}]_{m \times n} \quad (7.13)$$

$$t_{pq} = \begin{cases} \text{type value of C-NODE } r_p^c, & p = q \\ \text{type value of C-LINK } c_{pq}^c, & \text{otherwise} \end{cases} \quad (7.14)$$

As an example, Figure 7.1 shows SRAGs, R_1 and R_2 of radical samples  and . The corresponding radical adjacency matrices are A and B .

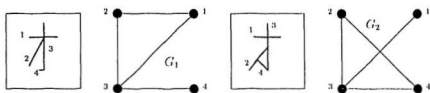




Figure 7.1 SRAGs of radical samples  and 

$$A = [a_{ij}] = \begin{pmatrix} 000101 & 001000 & 100000 & 000000 \\ 001000 & 001001 & 000100 & 000000 \\ 100000 & 000100 & 010001 & 000001 \\ 000000 & 000000 & 000001 & 000110 \end{pmatrix}$$

$$B = [b_{ij}] = \begin{pmatrix} 000101 & 000000 & 100000 & 000000 \\ 000000 & 001001 & 000100 & 000010 \\ 100000 & 000100 & 010001 & 000001 \\ 000000 & 000010 & 000001 & 100010 \end{pmatrix}$$

The reduced skeleton graphs R_r of R_1 and R_2 are given in Figure 7.2.

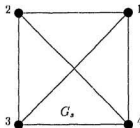


Figure 7.2 Reduced skeleton graph R_r of SRAGs R_1 and R_2

The corresponding incidence matrix T is

$$T = [t_{ij}] = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

where $t_{ij} = a_{ii} \odot_v b_{ii}$, and $t'_{ij} = a_{ij} \odot_e b_{ij}$ ($i \neq j$). For example, $t_{11} = a_{11} \odot_v b_{11} = 1$ and $t_{12} = a_{12} \odot_e b_{12} = 1$. All other t_{ij} can be obtained in a similar way. Here, $\text{order}(R_r) = \text{order}(R_1) = \text{order}(R_2) = 4$. Therefore, R_1 and R_2 can be synthesized. In fact, both radical samples represented by R_1 and R_2 are two variations of radical



7.2.2 Synthesis evaluation of two HAGs

Assume that two HAGs are $H_1 = (X_1, A_1)$ and $H_2 = (X_2, A_2)$, where $X_1 = \{V_1, \dots, V_m\}$, $X_2 = \{V'_1, \dots, V'_m\}$, $A_1 = \{\dots, E_{ij}, \dots\}$, and $A_2 = \{\dots, E'_{gh}, \dots\}$. The character ad-

jacency matrices corresponding to H_1 and H_2 are $D = [d_{ij}]_{m \times m}$ and $F = [f_{gh}]_{m \times m}$.

The corresponding skeleton graph is $H_s = (X_s, A_s)$.

The comparison result of two vertices in two HAGs could be: exactly match(e-match) or unmatched(u-match). An *e-match* indicates that two vertices V_i and V'_g match such that $d_{ii} = f_{gg}$. An *u-match* indicates that two vertices V_i and V'_g do not match such that $d_{ii} \neq f_{gg}$. For two given vertices V_i and V'_g , the type value of the corresponding C-NODE V_i^c ($V_i^c \in X_s$) is defined such that:

$$TYPE(V_i^c) = \begin{cases} 1, & \text{if } V_i \text{ and } V_g \text{ are e-match} \\ 0, & \text{if } V_i \text{ and } V_g \text{ are u-match} \end{cases} \quad (7.15)$$

The comparison result of two arcs in two HAGs could be exactly match(e-match) or unmatched(u-match): An *e-match* shows that both arcs E_{ij} and E'_{gh} match such that $d_{ij} = f_{gh}$. An *u-match* shows that both arcs E_{ij} and E'_{gh} mismatch such that $d_{ij} \neq f_{gh}$. For two given arcs E_{ij} and E'_{gh} , the type value of corresponding C-LINK E_{ij}^c ($E_{ij}^c \in A_s$) is defined such that:

$$TYPE(E_{ij}^c) = \begin{cases} 1, & \text{if } E_{ij} \text{ and } E_{gh} \text{ are e-match} \\ 0, & \text{if } E_{ij} \text{ and } E_{gh} \text{ are u-match} \end{cases} \quad (7.16)$$

The algorithm to reduce the skeleton graph is :

1. Remove the C-LINKs and C-NODEs with type value equal to 0.
2. Delete the C-NODEs connected to the removed C-LINKs.
3. Delete the C-LINKs incident in or out of the removed C-NODEs.

Definition 7.4 A reduced skeleton graph H_s of two HAGs H_1 and H_2 with the same number of vertices is called completion of H_1 and H_2 if $\text{order}(H_s) = \text{order}(H_1)$.

The condition that two HAGs H_1 and H_2 can be synthesized is:

1. The completion of H_1 and H_2 exists.
2. For each C-NODE in the completion, the completion of the two RAGs corresponding to the vertices described by the C-NODE exists.

7.2.3 Synthesis of two radical attributed graphs

If completion G_c of two original radical graphs R_1 and R_2 is found, a graph synthesis operation \cup_r can be applied to integrate them into a new radical graph R_n called convergent radical graph. Suppose that $R_1 = (V_1, E_1)$, $R_2 = (V_2, E_2)$, and $R_n = (V_n, E_n)$. $A = [a_{ij}]_{n \times n}$, $B = [b_{st}]_{n \times n}$, and $D = [d_{kt}]_{n \times n}$ are corresponding radical adjacency matrices. $G_n = R_1 \cup_r R_2$ if:

1. For each C-NODE in G_c , which records the comparison of vertices $v_f \in V_1$ and $v'_g \in V_2$, there is a vertex $v''_j \in V_n$ such that $d_{ff} = a_{ff} \oplus b_{gg}$.
2. For each C-LINK that annotates the comparison of edges $c_{ij} \in E_1$ and $c'_{st} \in E_2$, there is an edge $c''_{ij} \in E_n$ such that $d_{ij} = a_{ij} \oplus b_{st}$.

Here \oplus is "bit-wise or" operator.

For example, the synthesis of two radical graphs in Figure 7.1 is shown in Figure

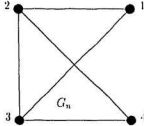


Figure 7.3 Convergent radical graph G_n of radical graphs R_1 and R_2 .

The corresponding radical adjacency matrix is:

$$D = [d_{ij}] = \begin{pmatrix} 000101 & 001000 & 100000 & 000000 \\ 001000 & 001001 & 000100 & 000010 \\ 100000 & 000100 & 010001 & 000001 \\ 000000 & 000010 & 000001 & 100111 \end{pmatrix}$$

7.2.4 Synthesis of two hierarchical attributed graphs

Similarly, when the completion H_c of H_1 and H_2 is found, a graph synthesis operation \cup_h can be used to unify H_1 and H_2 into a new graph H_n called a convergent hierarchical attributed graph. Let $H_1 = (X_1, A_1)$, $H_2 = (X_2, A_2)$, and $H_n = (X_n, A_n)$ and $P = [p_{ij}]$, $Q = [q_{kl}]$, $R = [r_{st}]$ are corresponding character adjacency matrices. R_1 is the RAG represented by $V_i \in X_1$, R'_k is the RAG represented by $V'_k \in X_2$, and R''_g is the RAG corresponding to $V''_g \in X_n$. $H_n = H_1 \cup_h H_2$ if:

1. Corresponding to each C-NODE that describes the comparison of vertices V_i and V'_j , there is a vertex $V''_t \in X_n$ such that $r_{ti} = p_{ji}$ and $R''_t = R_i \cup_r R'_j$.

2. Corresponding to each C-LINK which is an annotation of arcs E_{ij} and E_{kl} , there is an arc $b''_{ij} \in A_n$ such that $r_{ij} = p_{ij}$.

7.2.5 Graph synthesis by ordering

In order to determine whether two graphs can be integrated, the completion of the graphs should be derived. In the worst case all possible skeleton graphs for the two graphs should be obtained. It is prohibitively expensive to generate all the skeleton graphs. This is actually a combinatorial problem. To avoid the combinatorial explosion, the vertices in the HAGs or RAGs are ordered according to similar principles given in Chapter 5. For instance, the task of deriving the completion of two radical graphs can be accomplished by comparing correspond vertices or edges in the two graphs $R_1 = (V_1, E_1)$ and $R_2 = (V_2, E_2)$ in such a way that $v_i \in V_1$ is compared with $v'_i \in V_2$ and $e_{ij} \in E_1$ is compared with $e'_{ij} \in E_2$. With adjacency matrices, the synthesis evaluation of the two graphs becomes the comparison of corresponding entries in the adjacency matrices of the graphs. The result of the comparison is recorded in a skeleton incidence matrix. Obtaining the completion of two hierarchical attributed graphs can be done in a similar way.

For two radical graphs R_1 and R_2 with $A_{n \times n} = [a_{ij}]$ and $B_{n \times n} = [b_{ij}]$ as their radical adjacency matrices, their skeleton incidence matrix $T_{l \times l} = [t_{ij}]$ can be calculated

as:

$$t_{ij} = \begin{cases} a_{ii} \odot_v b_{ii} & \text{if } i = j \\ a_{ij} \odot_e b_{ij} & \text{if } i \neq j \end{cases} \quad (7.17)$$

The conditions that define whether R_1 and R_2 can be synthesized are:

- $l = n$.
- $t_{ij} = 1, \forall i, j, i \leq l$ and $j \leq l$.

The radical adjacency matrix $D_{l \times l} = [d_{ij}]$ of the new radical graph (convergent radical graph) R_n for R_1 and R_2 is:

$$d_{ij} = a_{ij} \odot b_{ij}. \quad (7.18)$$

For two HAGs $H_1 = (X_1, A_1)$ and $H_2 = (X_2, A_2)$ with $P_{m \times m} = [p_{ij}]$ and $Q_{m \times m} = [q_{ij}]$, their skeleton incidence matrix $T''_{s \times s} = [t'_{ij}]$ can be calculated as:

$$t'_{ij} = \begin{cases} 1, & \text{if } p_{ij} = q_{ij} \\ 0, & \text{if } p_{ij} \neq q_{ij} \end{cases} \quad (7.19)$$




The conditions that determine whether H_1 and H_2 can be synthesized are:

- $s = m$.
- $t'_{ij} = 1, \forall i, j, i \leq s$ and $j \leq s$.
- For each $1 \leq i \leq s$, the radical graphs R_1 for vertex $V_i \in X_1$ and the radical graph R'_1 for vertex $V'_i \in X_2$ can be synthesized.

The character adjacency matrix of the new H_n is $F_{s \times s} = [f_{ij}]$ where $f_{ij} = p_{ij}$.

7.3 Update the model database with the samples

The samples of an input character (a character will be learned) are well selected by a teacher. Only positive samples are learned by the system. If there exists a model of the input character in the database, it may be modified to include the samples with graph synthesis. In order to determine whether a model of the input character exists in the database, the tree has to be searched with the SHAGs of these samples. If there is no model matched by any SHAG of samples, the SHAGs will be inserted in the tree as new models.

It is possible that a SHAG of an input character and the MHAG of the character are not matched by calculating the mapping cost. Consider a single radical character  as an example. Here,  and  are its model and sample (see Figure 7.1). A and B are the corresponding adjacency matrices. The corresponding Mapping Cost Function (MMC) for A and B is:

$$MMC(A, B) = 2$$



In this case, the model can not be found and the SHAG will be inserted in the tree as a new model. This will introduce redundancy in the model database and consequently increase the size of the model database.

Definition 7.5 For a character with several radicals, any other character that shares at least one or more common radicals is defined as *co-radical character* of this character.

Definition 7.6 For a radical, any character which consists of this radical is defined as *radical-derived character* of the radical.

Definition 7.7 A radical is defined as *common radical* of its radical-derived characters.

Definition 7.8 A radical is defined as *learned radical*, if a model of the radical exists in the model database.

Moreover, the inconsistency for common radicals of radical-derived characters would be produced in the database. For the same example, all radical-derived characters in the database of radical  have to be updated to include the new SRAG for . It is very difficult to find all these radical-derived characters in the tree without traversing every node in the tree.

Instead of tree searching, several new strategies are proposed to overcome this difficulty:

- **Sample acquisition.** During the learning phase, the teacher should provide the character code and the radical codes of component radicals for the input character.
- **Table network organization.** This provides direct and fast accessing and updating of the database.
- **Integrating algorithm.** Use graph synthesis to update the model database so that the inconsistencies and the redundancies can be avoided.

7.3.1 Table network organization of model database

The network consists of two hosts: the host for characters and the host for radicals. Each host consists of several functional tables.

The host for characters

At the host for characters, there are three kinds of tables: character table, MHAG subtables, and component RAG subtables.

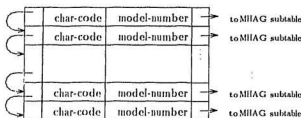


Figure 7.4 Character table.

Character table (Figure 7.4) is a linked list which stores the models of characters. Each record in the character table consists of three fields: *char-code*, *number of models*, and *pointer*. *Char-code* holds the character code of the corresponding character; *pointer* points to the MHAG subtable of the character. The character is searched sequentially.

Following the dashed line starting at MHAGsubtable in Figure 7.6, the internal structure of the MHAG subtable of a character is illustrated. The MHAG subtable is a linked list. Each MHAG for a model is stored as a record in the MHAG subtable. Each record consists of three fields: *number of radicals*, *CAM*, and a *pointer*. *CAM* is the character adjacency matrix of the model; *pointer* points to the component RAG subtable of the model.

Following the dashed line starting at **CRAGsubtable** in Figure 7.6, the internal structure of the component RAG subtable of a model is illustrated. The component RAG subtable is a one dimensional array and the size of the array is decided by the number of the radicals in the model. In the RAG subtable, each record corresponding to a component radical has two fields: *radical-code* and *address of RAG*. The RAG of the component radical is stored at the host for radicals.

The host for radicals

At the host for radicals, there are three kinds of tables: radical table, index subtables, and MRAG subtables. These tables store the information about all the radicals in the database.

Similar to the character table, the radical table is also a linked list. In the radical table, each record corresponding to a radical, consists of three fields: *radical-code*, and two *pointers*. One *pointer* points to an index subtable, while the other points to a MRAG subtable. Figure 7.5 shows the structure of a radical table.

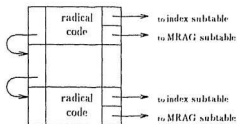


Figure 7.5 Radical table.

After the dashed line from **MRAG subtable** in Figure 7.6, the internal structure

of the MRAG subtable is illustrated. The MRAG subtable is a linked list. Each MRAG of a radical is described by a record in the table. Each record has two fields: *number of strokes* and *RAM*. Following the dashed line starting at index subtable in Figure 7.6, the internal structure of the index subtable is shown. The index subtable is also a linked list. Records of the subtable consist of the char-codes for all the radical-derived characters.

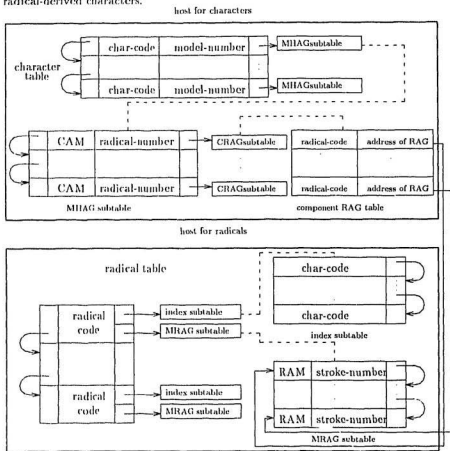


Figure 7.6 Table network organization of model database.

Figure 7.6 illustrates the overall table network organization of model database. The dashed lines connect subtables and their internal structures of MHAG subtables, component RAG subtables, index subtables, and MRAG subtables in the network.

7.3.2 RAG integration and HAG integration

RAG integration and HAG integration are proposed to synthesize radical attributed graphs and hierarchical attributed graphs. Both integrations are the fundamental functional components used for learning by the samples of an input character.

The RAGs to be synthesized are stored in a list called the RAG integration list. Each element of the list has two fields: the number of strokes and RAM. If two RAGs in the RAG integration list can be synthesized, they are replaced by the convergent RAG. This process is continued until there are no two RAGs that can be integrated in the RAG integration list.

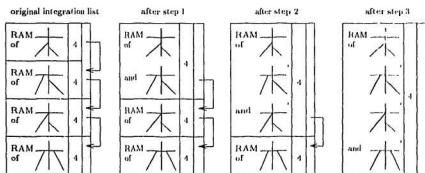


Figure 7.7 Example of RAG integration.

Figure 7.7 illustrates an example of RAG integration. Initially, there are four

elements representing four RAGs of a radical in the RAG integration list. With the graph synthesis evaluation, the RAG of the first element can be synthesized with that of the second one. After the first step, both RAGs are replaced by a new RAG in the integration list, and now there are only three records in the list. In a similar way, this process continues.

The HAG integration is designed to synthesize the hierarchical attributed graphs. The hierarchical attributed graphs (HAGs) are stored in the HAG integration structure which consists of a HAG integration subtable, component RAG subtables, and RAG integration lists. A HAG integration subtable is similar to a MHAG subtable. The record of a HAG in the HAG integration table consists of three fields: number of radicals, CAM, and a pointer, which points to the component RAG subtable of the HAG. Each record of the component RAG subtable consists of two fields: a radical-code and an address of RAG which is stored in a RAG integration list. Based on the synthesis conditions of two HAGs in subsection 7.2.5, it can be easily concluded that the conditions that two HAGs in a HAG integration subtable can be integrated are: (1) they have the same CAM, and (2) both of their component RAG subtables are the same. Under these conditions, both HAGs are replaced by their convergent HAG. Similarly, the process continues until no two HAGs can be synthesized in the HAG integration subtable.

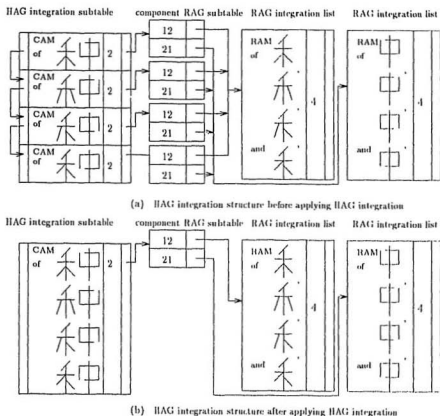


Figure 7.8 Example of HAG integration.

For example, Figure 7.8 shows how HAGs of an input character are integrated. Figure 7.8(a) illustrates the HAG integration structure before HAG integration. There are four records representing four HAGs of the character in the HAG integration table. Figure 7.8(b) shows the HAG integration structure after HAG integration.

7.3.3 Modification of models

With the table network organization, RAG integration, and HAG integration, the model database can be updated directly and quickly by learning from samples. The modification procedure consists of two phases: transformation and updating.

Transformation

For each sample of an input character, there is a SHAG and several SRAGs which correspond to the radicals in the sample. In the transformation phase, integration of RAG and HAG is performed to reduce the number of SHAGs and SRAGs. For each component radical of the input character, its RAGs of the samples (SRAGs) are obtained and stored in its corresponding RAG integration list. Then the HAGs of the samples (SHAGs) are constructed and stored in its corresponding HAG integration structure. RAG integration is applied on each RAG integration list in the HAG integration structure such that no two SRAGs in the list can be further synthesized. At last, the HAG integration is applied on the HAG integration structure such that no two HAGs in the structure can be further synthesized. The steps for transformation of the input character are:

- Step 1.** For each component of the input character, create a RAG integration list to store its SRAGs.
- Step 2.** Create HAG integration structure to store the SHAGs of the character.

Step 3. Apply RAG integration to each RAG integration list and adjust the address of RAG in each component RAG subtable.

Step 4. Apply IIAG integration on the IIAG integration structure.

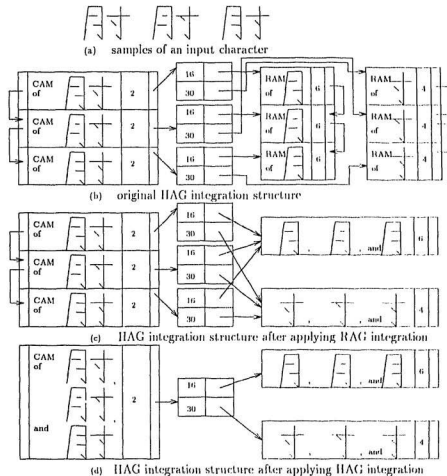


Figure 7.9 Transformation of samples for an input character.

Figure 7.9 illustrates an example of transformation. The input character has three samples (Figure 7.9(a)). The original IIAG integration structure of the input character before RAG integration and IIAG integration is given in Figure 7.9(b). Figure 7.9(c) shows two RAG integration tables after applying RAG integration. Figure 7.9(d) shows the IIAG integration structure after applying IIAG integration.

Updating

There are two cases of updating. The first case is that an input character has been learned before. The models for the character exist in the database and the change of the models is necessary. The second case is that an input character has never been learned, and there is therefore no model for the character in the model database. New models for the input character have to be inserted in the database. In both cases, for the learned radicals of the input character, the models of their radical-derived characters in the model database should be modified according to the RAG integration lists of these radicals.

Case 1. The input character has been learned before. All its component radicals are the learned radicals. The updating procedure for this case is:

Step 1. a) Make a copy of the MIIAG table of the found models from the host for characters.

b) Make a copy of each component RAG subtable corresponding to each model.

Step 2. For each component radical, make a copy for its MRAG subtable in the host

for radicals.

Step 3. For the input character, update the HAG integration structure obtained in the transformation phase.

- a) Append the copy of the MHAG table to the HAG integration subtable of the input character.
- b) Add the copies of the component RAG subtables corresponding to the models into the HAG integration structure.
- c) Append the copy of each MRAG table to the corresponding RAG integration list.

Step 4. a) Apply RAG integration to each RAG integration list and adjust the address of RAG in each component RAG subtable.

- b) Apply HAG integration to the HAG integration structure.

Step 5. a) Replace the MHAG subtable and the component RAG subtables of the input character at the host for characters with the HAG integration subtable and the component RAG subtables of the HAG integration structure.

- b) For each component radical, replace its MRAG subtable in the host for radicals with its RAG integration list.

Step 6. According to the radical code of each component radical which is a learned radical, obtain its index subtable through the radical table at the host for radi-

cals. In the radical table, there is a char-code for each radical-derived character of the radical.

Step 7. With each char-code, except for the char-code of the input character in the index subtable, search the character table to find the MHAG subtable of corresponding radical-derived character.

Step 8. Based on the MHAG subtable, obtain its component RAG subtables and adjust the addresses of RAG in these component RAG subtables.

Step 9. For the MHAG subtable of each radical-derived character, synthesize MHAGs if they have the same CAMs and component RAG subtables until there are no two MHAGs which can be synthesized in this way. Exist the updating.

Case 2. The input character has never been learned. There may be some component radicals which are learned radicals. The updating procedure for this case is:

Step 1. With the radical codes of the component radicals of the input character, search the radical table for the component radicals which are learned radicals.

Step 2. Make a copy for the MRAG subtable of each learned radical and append the copy to the corresponding RAG integration list of the radical in the HAG integration structure of the input character.

Step 3. a) In the HAG integration structure, apply RAG integration to the RAG integration list of each learned radical and adjust the address of RAG in the component RAG subtable.

- b) Apply IIAG integration to the IIAG integration structure.

Step 4. Insert new models of the input character into the database.

- a) Insert a record containing the char-code and model-number of the input character into the character table.
- b) Add the IIAG integration subtable and the component RAG subtables of IIAG integration structure in the host for character as the MIIAG subtable and the component RAG subtables for the input character.

Step 5. For each component radical which is a learned radical, replace its MRAG subtable with its RAG integration list and insert the char-code of the input character in the corresponding index subtable as a new radical-derived character.

Step 6. For each component radical which is not a learned radical, insert a record with the radical code of the radical in the radical table, build an index subtable which contains the char-code of the input character as a new radical-derived character, and add the corresponding RAG integration list into the host for radicals as its MRAG subtable.

Step 7-10. Same as steps 6-9, inclusive in case 1.

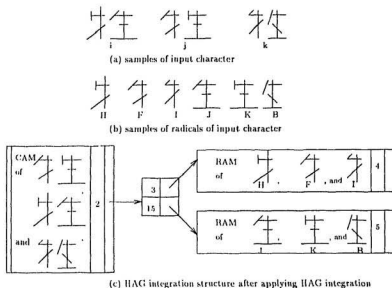


Figure 7.10 HAG integration structure of an input character.

As case 2 is easier and similar to case 1, an example is illustrated only for case 1. Three samples of an input character (推) and the corresponding HAG integration structure, after applying RAG integration and HAG integration, are shown in Figure 7.10.

Figure 7.11 illustrates the table network organization of the model database before updating. There are three co-radical characters (星, 姓, 北) of the input character in the table network, as well as two models of the input character, which can not be synthesized. As shown in Figure 7.11, the MHAG subtable of these two models, and the corresponding component RAG subtables and MRAG subtable are indicated with rectangles of bold lines.

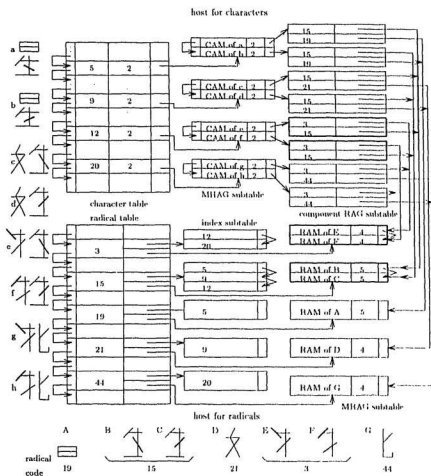


Figure 7.11 Table network organization of database.

Figure 7.12(a) illustrates the HAG integration structure after step 3. Figure 7.12(b) indicates the HAG integration structure after applying RAG integration and HAG integration of step 4.

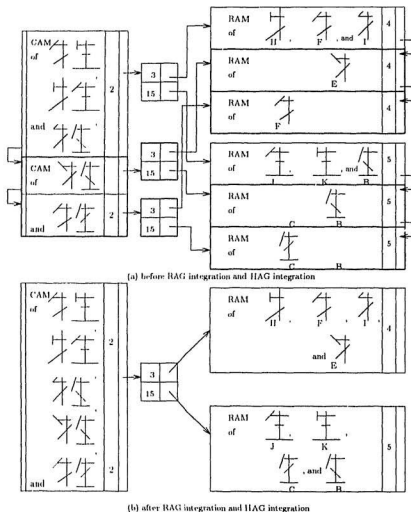


Figure 7.12 RAG integration and HAG integration in the updating.

Figure 7.13 illustrates the result of updating the table network in Figure 7.11 after step 8. The MIRAG subtable, the corresponding component RAG subtables,

and the corresponding MRAG subtables of the input character in the table network are replaced by the those subtables in the HAG integration structure. These new subtables are indicated by rectangles of bold lines in Figure 7.13. The addresses of RAG in the component RAG tables of the co-radical characters of the input character are adjusted. These component RAG tables are marked by rectangles of bold dashed lines in Figure 7.13.

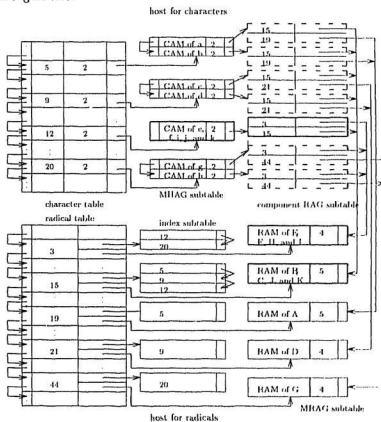


Figure 7.13 Updating results after step 8.

In step 9, the HAGs in each MHAG subtable with the same component RAG

subtables are synthesized. The updated MHAG subtables are indicated by rectangles of bold lines in Figure 7.14.

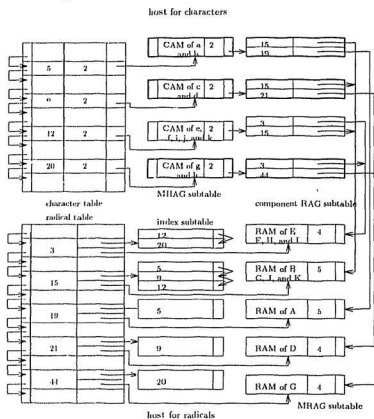


Figure 7.14 Final updating results for the input character.

After the learning process, the table network contains the information of the models for learned characters. After the database is updated, a new multi-way heterogeneous tree can be built for recognition according to the table network. All the models are inserted into the tree, one by one, according to the algorithm in section 6.3.

Chapter 8

Conclusions

8.1 Summary of contributions

In this thesis, we have presented a new method for effective recognition of handwritten Chinese characters. A structural representation, called hierarchical attributed graph representation (HAGR), is introduced to represent and describe both invariant and unstable features. First, the strokes are extracted and grouped into radicals. After that, the radical attributed graph $G_r = (V_r, E_r)$ is constructed. The adjacency matrix has been chosen as the data structure of the attributed graph in the present system. The bits of the entries in the matrix describe the attribute set associated with the vertex or edge. There are three advantages to using bits of an entry to represent an attributed set: (1) storage is reduced, (2) graph matching efficiency is improved, and (3) the variation related to the stroke type or orientation is allowed without having to

increase the number of models for each character in the model database. An HAGR for the character can be constructed based on the radicals, thus the recognition process becomes a simple task of graph matching. A cost function, mapping a candidate to a model graph, is introduced to measure the matching of graphs. With a matching procedure, characters can be recognized if a match is found between the HAGR of a character and a stored model. This approach can tolerate the variations of HAGR which reflect the instabilities or variabilities of handwritten Chinese characters resulting from different writing styles. Characters with different writing styles can be correctly recognized as the same characters. HAGR and graph matching offer an efficient method for Chinese character recognition.

In order to achieve accuracy and speed in the recognition of handwritten Chinese characters, the database of the character models is organized as a multi-way tree structure. To avoid memory unit waste, a radical linked list is built to store the radical adjacency matrices, and index arrays and compression vectors are employed with the tree structure to organize the database. Instead of duplicating matrices in the tree, the address of a *RAM* in the radical linked list is stored in the tree. With the tree structure, the difficult and tedious searching of the model database to find a model character to match an input character can be broken up into a number of simple and local decisions at different levels of the tree. The unmatched models in the model database can therefore be excluded at an early stage of searching and the searching time can be greatly reduced.

A learning algorithm using graph synthesis has been developed to create and update models in the model database based on the hierarchical attributed graphs extracted from the samples of input characters. This algorithm is capable of (1) synthesizing the graphs among the samples of an input character, and (2) modifying the graphs of existing models of the input character using the graphs extracted and synthesized from these samples. With the learning algorithm, only a few models need to be built for a character instead of using many models corresponding to each sample of the character. The model database consists of 660 different character classes which cover most of the typical structures of Chinese characters. Each class has on average, five different models. In total, 3300 character models are created by the learning algorithm in the database.

8.2 Experiment and analysis

The system has been implemented in C and experiments of character recognition have been conducted on a MIPS/M-120 running RISC/OS (Version 3.1) using test data consisting of sixty character sets written by five people. Each set contains up to ten different writing styles with variations in strokes and stroke connections such as strokes with different lengths, certain deviations of stroke direction, various stroke connections, and stroke types (e.g. dot or line), which differ according to different writing habits. The recognition rate can still be maintained at over 92%. The rejection rate is 7% and the misclassification rate is much less than 0.01%. The

average time needed to search the multi-way tree for an input character is about 0.065s cpu time. Figure 8.1 shows some of the samples of handwritten characters recognized by this system. The searching time for single component characters is shorter than that for compound characters. This is reasonable as more radicals have to be checked for compound characters.

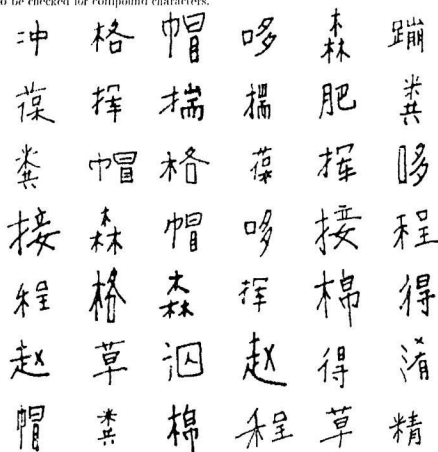


Figure 8.1. Recognized samples.

The results achieved by the proposed method in this experiment are very promising.

(1) Some Chinese characters such as (甲, 申, 由), (己, 巳), (右, 古), (叮, 可), and (太, 大, 大) are considered difficult to identify by existing techniques [Suen 1986], but they are correctly recognized by our algorithm with no difficulty.

(2) Characters with different writing styles such as (韦, 韋), (隹, 隹), (男, 男), (向, 多, 多), (口, 口), (斗, 斗), (禾, 禾), (文, 文), (木, 木), and (古, 古) can be correctly recognized as the same characters.

The reasons for these encouraging results are analyzed and highlighted below. The above groups can be divided into five classes.

Class A: In groups (甲, 申, 由), (己, 巳), (右, 古), and (太, 大, 大), the different connections, or numbers of strokes for characters of each group with similar configurations are described differently in their HAGRs and hence they can be distinguished from each other. For instance in group (甲, 申, 由), the connection between the top horizontal stroke and the middle vertical stroke in the first character represented as T is different from the connection in the second character or the third character which has no relation T.

Class B: In group (叮, 可), the spatial relation between radicals in the first character is LR, while that in the second character is BC.

Class C: In the groups (斗, 斗), (禾, 禾), (文, 文), (木, 木), (古, 古), and (口, 口), the characters written with different connections between strokes

and different stroke types can still be identified with our method since the HAGRs of the models contain the variations of these characters.

(Class D: In the groups $(\begin{smallmatrix} \pm \\ \downarrow \end{smallmatrix}, \begin{smallmatrix} \pm \\ \uparrow \end{smallmatrix})$, $(\begin{smallmatrix} \pm \\ \downarrow \end{smallmatrix}, \begin{smallmatrix} \pm \\ \uparrow \end{smallmatrix})$, $(\begin{smallmatrix} \pm \\ \downarrow \end{smallmatrix}, \begin{smallmatrix} \pm \\ \uparrow \end{smallmatrix})$, and $(\begin{smallmatrix} \pm \\ \downarrow \end{smallmatrix}, \begin{smallmatrix} \pm \\ \uparrow \end{smallmatrix})$, the variations of the same character are written in different structures such as different number or different configurations of radicals. Because such variations can not be represented by a single HAGR model, we use multiple HAGR models in the database to represent them. Therefore, they can still be recognized as the same character by the system. Due to the multi-way tree organization of the model database, and the radical link list, the amount of memory required for the multiple models does not increase much with comparisons to a single model. For example, the radical attributed graphs of radicals $\begin{smallmatrix} \pm \\ \downarrow \end{smallmatrix}$, $\begin{smallmatrix} \pm \\ \uparrow \end{smallmatrix}$, $\begin{smallmatrix} \pm \\ \downarrow \end{smallmatrix}$, $\begin{smallmatrix} \pm \\ \uparrow \end{smallmatrix}$, and \square are uniquely stored in the radical link list without duplications.

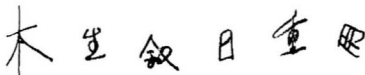


Figure 8.2. Character samples being rejected or misclassified.

Figure 8.2. shows some of the samples being rejected or misclassified. Rejection or misclassification are caused by too much rotation, the perfunctory nature of strokes, or the cursive connection of strokes. If an input character cannot be recognized, it will be rejected very quickly, as this decision can be made at any level in the tree

without searching all the models in database.

8.3 Directions for future research

Directly related to the research in the thesis, there are several interesting problems that need further consideration.

8.3.1 Radical grouping

An algorithm to group strokes into radicals has been proposed based on the connective properties of strokes and box properties of radicals. However, for a character with several radicals, a radical is not always separated from other radicals and may have one or more strokes connecting with the strokes of other radicals due to careless writing or noise effects. In this situation, the proposed algorithm fails to separate radicals touching each other in a character. One possible way to extract the unseparated radical is to find a monomorphism between the model graph of the radical and the graph of the part containing the unseparated graph under certain constraints.

8.3.2 Error tolerating matching

Sometimes, a character is written with several different numbers of strokes but its shape and major configuration remain unchanged. In order to recognize all these variations as the same character, several models are required. If an error-tolerant matching algorithm which can ignore such minute differences can be developed, the

number of models required to represent character variations can be greatly reduced.

8.3.3 Learning from feedback

With the learning algorithm currently implemented in our system, the character samples to be learned are controlled and selected by human teachers. The quality of the samples depends heavily on the standards of these teachers. The freedom provided by such a learning process is very limited. For improvement, a learning algorithm based on feedback of the system from the recognition phase can be implemented in the future. For an input character that cannot be matched by any model in the database, the frequency of the rejection of that character is counted as feedback. When the frequency is over a given threshold value, the character will be automatically selected as a character sample. The system will then expand the database to include the HAGR of this character as a new model or use this HAGR to update the model of an existing character.

References

- Aho, A. V., J. E. Hopcroft, and J. D. Ullman [1974]. *The Design and Analysis of Computer Algorithms*, Addison-Wesley, pp. 124-164.
- Akamatus S. and K. Komori [1981]. "Concentrated structural features for handprinted Chinese recognition", *Trans. IECE Japan*, vol. PRL80-83.
- Arakawa H. [1983]. "On-line recognition of handwritten characters Alphanumerics, Hiragana, Katakana, Kanji", *Pattern Recognition*, vol. 16, no. 1, pp. 9-16.
- Arakawa H., K. Odaka, and I. Masuda [1978]. "On-line recognition of handwritten characters - Alphanumeric, Hiragana, Katakana, Kanji", *Proc. 4th Int. Joint Conf. Pattern Recognition*, pp. 810-812.
- Casey R. and G. Nagy [1966]. "Recognition of printed Chinese characters", *IEEE Trans. Electron. comput.*, vol. EC-15, pp. 91-101.
- Chan K. P. and Y. S. Cheung [1989]. "Fuzzy-attributed graph and its application to Chinese character recognition", *Computer Processing of Chinese & Oriental Languages*, vol. 4, pp. 85-98.
- Chang S. K. and D. H. Lo [1973]. "An experimental system for the recognition of handwritten Chinese characters", *Proc. 1st Int. Symp. Comput. and Chinese Input-Output Syst.*, pp. 257-267.
- Chen K. J., K. C. Li, and Y. L. Chang [1988]. "A system for on-line recognition of Chinese characters", *Comput. Processing of Chinese & Oriental Languages*, vol. 3, No. 3 &

4, pp. 309-318.

Chen P. N., Y. S. Chen, and W. H. Hsu [1988]. "Stroke relation coding - A new approach to the recognition of multi-font printed Chinese characters", *Computer Processing of Chinese & Oriental Languages*, vol. 3, nos. 3 & 4, pp. 319-330.

Cheng F. H., W. H. Hsu, and M. Y. Chen [1989]. "Recognition of handwritten Chinese characters by modified Hough transform techniques", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-11, No. 4, pp. 429-439.

Cox C. H., P. Conignoux, B. Blesser, and M. Eden [1982]. "Skeletons: A line between theoretical and physical letter descriptions", *Pattern Recognition*, vol. 15, pp. 11-22.

Crane H. D. and R. e. Savoie [1977]. "An on-line data entry system for handprinted characters", *Computer*, vol. 10, pp. 43-50.

Dai Y., N. N. Zheng, X. N. Zhang, and G. R. Xuan [1988]. "Automatic recognition of province name on the license plate of moving vehicle", *Proc. 9th Int. Conf. Pattern Recognition*, pp. 927-929.

Engdahl J. [1977]. "Data entry and decoding system for scripted data", U. S. Patent 4 005 400.

Farag R. F. [1979]. "Word-level recognition of cursive script", *IEEE Trans. comput.*, vol. C-28, pp. 171-175.

Fu K. S. [1982]. *Syntactic Pattern Recognition and Application*, Prentice Hall.

Fujii N., H. Sugawara, E. Yamamoto, C. Ito, and T. Fujita [1981]. "Some results on handprinted Kanji character recognition using the feature extracted from multiple standpoint", *Trans. IECE Japan*, vol. PRL81-32.

Glucksman H. A. [1967]. "Classification of mixed font alphabetic by characteristic loci", *Dig. 1st Annu. IEEE Comput. Conf.*, pp. 137-141.

Greenias E. C. and E. F. Yhap [1982]. "Chinese/Kanji on-line recognition system", U. S. Patent 4 365 235.

- Groner G. F. [1968]. "Real-time recognition of handprinted symbols", in L. N. Kanal, Ed., *Pattern Recognition*. Washington, DC: Thompson, pp. 103-108.
- Gu Y. X., Q. R. Wang, and C. Y. Suen [1983]. "Application of a multi-layer decision tree in computer recognition of Chinese characters", *IEEE Trans. Pattern Ana. & Machine Intel.*, vol. PAMI-5, No 1, pp. 83-89.
- Hagita N. and I. Masuda [1981]. "Handprinted Chinese character recognition", *Trans. IECE*, vol. PRL81-13.
- Hanaki S., T. Temma, and H. Yoshida [1976]. "An on-line recognition aimed at a substitution for a billing machine keyboard", *Pattern Recognition*, vol. 8, pp. 63-71.
- Hanaki S. and T. Yamazaki [1980]. "On-line recognition of handprinted Kanji characters", *Pattern Recognition*, vol. 12, pp. 421-429.
- Hing-hua R. [1988]. "a practical recognition system for inputting handwritten Chinese characters on-line", *Proc. 1988 Int. Conf. Comput. Processing of Chinese and Oriental Languages*, pp. 62-64.
- Ikeda K., T. Yamanura, Y. Mitamura, S. Fujiwara, Y. Tominaga, and T. Kiyono [1978]. "On-line recognition of handwritten characters utilizing positional and stroke vector sequences", *Proc. 4th Int. Joint Conf. Pattern Recognition*, pp. 813-815.
- Impedovo S. [1984]. "Plane curve classification through Fourier descriptors: an application to Arabic hand-written numeral Recognition", *Proc. 7th Int. Conf. Pattern Recognition*, pp. 1069-1072.
- Ishigaki K. and T. Morishita [1988]. "A top-down online handwritten character recognition method via the denotation of variation", *Proc. 1988 Int. Conf. Comput. Processing of Chinese and Oriental Languages*, pp. 141-145.
- Ishii Y. [1986]. "Stroke order free on-line handwritten Kanji character recognition method by means of stroke representation points", *Trans. Inst. Electron. Commun. Eng. Japan*, pp. 1069-1072.

- Kasvand T. [1979]. "Ideogram segmentation and recognition and recognition", *Proc. Int. Conf. Cybern. Soc.*, pp. 674-678.
- Kato O., T. Fujita, M. Niwa, T. Morishita, and J. Tanahashi [1978]. "A handwritten input system for Japanese", *Proc. IFIP*, pp. 689-694.
- Kerrick D. D. and A. C. Bovik [1988]. "Microprocessor-based recognition of handprinted characters from a tablet input", *Pattern Recognition*, vol. 21, pp. 525-537.
- Kimura F., M. Yoshimura, Y. Miyake, and M. Ichikawa [1978]. "Undeterministic stroke extraction method for skeletonized character patterns", *J. IECE Japan*, vol. J61-D, pp. 496-503.
- Krzyzak A. and H. F. Buaeshi [1989]. "Classification of digitized curves represented by signatures and Fourier descriptors", *Computer vision and shape recognition*, World Scientific, pp. 241-259.
- Kuo H. W., J. C. Lee, and T. C. Kao [1988]. "IGLCR, An on-line Chinese character recognition system", *Proc. 1988 Int. Conf. Comput. Processing of Chinese and Oriental Languages*, pp. 108-112.
- Lee S. W. and J. M. Kim [1989]. "Automatic entity verification of seal imprint", *Comput. Process. Chinese and Oriental Lang.*, vol. 4, pp. 124-141.
- Lee S. Y. and K. S. Fu [1977]. "Stochastic Error-correcting syntax analysis for recognition of noisy patterns", *IEEE Trans. Syst. Man. and Cybern.*, vol. SMC-7, No. 12, pp. 1268-1276.
- Leow W. K. [1987]. "Research in Chinese character recognition", *Proc. of Chinese Computing Seminar '87*, Singapore, pp. 81-100.
- Leung C. H. [1985]. "A practical basis set for Chinese character representation", *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 532-537.
- Li C. C., T. L. Teng, M. J. Zobrak, and T. W. Sze [1967]. "On recognition of handwriting Chinese characters", *Proc. 1st Princeton Conf. Inform. Sci. Syst.*, pp. 235-239.

- Liao C. W. and J. S. Huang [1990]. "Stroke segmentation by Bernstein-Bezier curve fitting", *Pattern Recognition*, vol. 23, no. 5, pp. 475-484.
- Lin M. Y. and W. H. Tsai [1988]. "A new approach to on-line Chinese character recognition by sentence contextual information using relaxation technique", *Proc. 1988 Int. Conf. Comput. Processing of Chinese and Oriental Languages*, pp. 62-64.
- Lu. H. E and P. S. P. Wang [1985]. "A fast parallel algorithm for thinning digital patterns", *Commun. ACM*, vol. 29, no. 3, pp. 239-242.
- Lu S. W., Y. Ren, and C. Y. Suen [1990]. "Handwritten Chinese character recognition with attributed graph matching", *Proc. Int. Vision Interface/Graphic Interface '90*, pp. 186-193.
- Mori S., T. Mori, K. Yamamoto, H. Yamada, T. Saito, and K. Nakata [1974]. "Field effect method", *Proc. 2nd Int. conf. Pattern Recognition*, pp. 233-237.
- Morishita T., M. Ooura and Y. Ishii [1988]. "A Kanji recognition method which detects writing errors", *Comput. Processing of Chinese & Oriental Languages*, vol. 3, No. 3 & 4, pp. 351-365.
- Naccache N. J. and R. Shinghal [1984]. "SPTA A proposed algorithm for thinning binary patterns", *IEEE Trans. Syst., Man., and Cybern.*, vol SMC-14, no. 3, pp. 409-418.
- Nagy G. [1988]. "Chinese character recognition: a twenty-five year retrospective", *Proc. Int. Conf. Pattern Recognition*, pp. 163-167.
- Naito S., K. Komori, and E. Yodogawa [1981]. "Stroke density feature for handprinted Chinese character recognition", *J. IECE Japan*, vol. J64-D, pp. 757-764.
- Nakata K., Y. Nakano, and Y. Uchikura [1972]. "Recognition of Chinese characters", *Proc. Conf. Machine Perception of Patterns of Pictures*, Teddington, pp. 45-52.
- Odaka K., H. Arakawa, and I. Masuda [1982]. "On-line recognition of handwritten characters by approximating each stroke with several points", *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-12, pp. 898-903.

- Odaka K., T. Wakahara, and I. Masuda [1986]. "Stroke-order-independent on-line character recognition algorithm and its application", *Rev. Elec. Commun. Lab.*, vol. 34, pp. 79-85.
- Oka R. [1982]. "Handwritten Chinese character recognition by using cellular feature", *Proc. 6th Int. Joint Conf. Pattern Recognition*, pp. 783-785.
- Pavlidis T. [1982]. "Algorithms for graphics and image processing", *Computer Science Press Inc.*, Rockville MD, pp. 195-214.
- Pavlidis T. [1982]. "An asynchronous thinning algorithm", *Comput. Graphics Image Processing*, vol. 20, pp. 133-157.
- Phamoudon R. and C. Y. Suen [1989]. "Thinning of digitized characters from subjective experiments: A proposal for a systematic evaluation protocol for algorithms" In press, *Computer Vision and shape Recognition*, eds. A. Krzyzak, T. Kasvand and C. Y. Suen, World Scientific Publishing Co. Ltd, Sing apore.
- Saito T., H. Yamada, and K. Yamamoto [1982]. "An analysis of handprinted Chinese characters", *J. IECE Japan*, vol. J65-D, pp. 550-557.
- Sakai K., S. Hirai, T. Kawada, S. Amano, and K. Mori [1976]. "An optical Chinese character reader", *Proc. 3rd Int. Joint Conf. Pattern Recognition*, pp. 122-126.
- Sakai T., K. Odaka, and T. Toida [1984]. "Several approaches to development of on-line handwritten character equipment", *Proc. 7th Int. Conf. Pattern Recognition*, pp. 1052-1054.
- Sato Y. and H. Adachi [1985]. "On-line recognition of cursive writings", *Trans. Inst. Electron. Commun. Eng. Japan*, vol. J68-D, pp. 2116-2112.
- Shi Q. Y. and K. S. Fu [1983]. "Parsing and Translation of (attributed) expensive graph languages for scenes analysis", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 472-485.
- Suen C. Y. [1982]. "The role of Multi-dimensional loci and clustering in reliable recognition of characters", *Proc. 6th Int. Conf. Pattern Recognition*, pp. 1020-1022.

- Tai J. W. [1984]. "A syntactic-semantic approach for Chinese character recognition", *Proc. 7th Int. Conf. Pattern Recognition*, pp. 374-376.
- Tai J. W. and Y. J. Liu [1990]. "Chinese character recognition", *Series in Computer Science*, World Scientific, vol. 7, pp. 415-451.
- Takahashi H. [1982]. "A simple recognition method for handprinted Kanji characters by using primitive connective directions of thinning", *Trans. IECE Japan*, vol. PR182/8, pp. 57-62.
- Tsai W. H. and K. S. Fu [1980] "A syntactic-statistic approach to recognition of industrial objects", *Proc. 5th Int. Conf. Pattern Recognition*, pp. 251-259.
- Tsai W. H. and K. S. Fu [1979] "Error-correction isomorphisms of attributed relational graphs for pattern analysis", *IEEE Trans. Syst. Man. and Cybern.*, vol. SMC-9, pp. 757-768.
- Verschueren W., B. Schaeken, Y. Rene De Cotret, A. Hermanne [1984]. "Structural recognition of handwritten numerals", *Proc. Int. Conf. Pattern Recognition*, pp. 760-762.
- Wakahara T. [1988]. "On-line cursive script recognition use local affine transformation", *Proc. 9th Int. Conf. Pattern Recognition*, pp. 1133-1137.
- Wakahara T. and M. Umeda [1983]. "Stroke-number and stroke-order free on line character recognition by selective stroke linkage method", *Proc. 4th IJTP*, pp. 157-162.
- Wakahara T. and M. Umeda [1984]. "On-line cursive script recognition using stroke linkage rules", *Proc. 7th Int. Conf. Pattern Recognition*, pp. 1065-1068.
- Wakayama T. [1982]. "A core-line tracing algorithm based on square moving", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, pp. 68-74.
- Wang P. S. P. [1988]. "Knowledge pattern representation of Chinese characters", *International Journal of Pattern Recognition and Artificial Intelligence*, vol 2, no. 1, pp. 161-179.

- Ward J. R. and M. J. Kuklinski [1988]. "A model for variability effects in handprinted with implications for the design of handwriting character recognition systems", *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 438-451.
- Watanabe Y., J. Gyoba, T. Hirata, and K. Maruyama [1985]. "A psychological approach to the human recognition of ambiguous characters", *J. Inst. TV Eng. Japan*, vol. 39, pp. 509-515.
- Yamamoto K., H. Yamada, T. Saito, and R. Oka [1984]. "Recognition of handprinted Chinese characters and Japanese cursive syllabary", *Proc. 7th Int. Conf. Pattern Recognition*, pp. 385-388.
- Yamashita Y. T., K. Higuchi, Y. Yamada and Y. Haga [1982]. "Classification of hand-printed Kanji characters by structured segment method", *Trans. IECE Japan* vol. PRL81-93.
- Yasuda M. and H. Fujisawa [1979]. "An improvement of correlation method for character recognition", *J. IECE Japan*, vol. J62-D, pp. 217-224.
- Yasuhara M. [1975]. "Experimental studies of handwriting process", *Rep. Lab. Commun. Sci., Univ. Electro-Commun., Japan*, vol. 25-2, pp.233-254.
- Yoshida K. and H. Sakoe [1982]. "On-line handwritten character recognition for a personal computer system", *IEEE Trans. Consumer Electron.*, vol. CE-28, pp. 202-209.
- Yoshida M. and M. Eden [1973]. "Handwritten Chinese character recognition by analysis-by-synthesis method", *Proc. 1st Int. Conf. Pattern Recognition*, pp. 197-204.
- Yurugi M., S. Nagata, K. Onuma, and K. Kubota [1985]. "On-line character recognition by hierarchical analysis method", *Trans. Inst. Electron. Commun. Japan*, vol. J68-D, pp. 1320-1327.
- Zhang S. and K. S. Fu [1984]. "A thinning algorithm for discrete binary images", *Proc. 1st Int. Conf. Comput. and Applications*, Beijing, pp. 879-886.
- Zhang T. Y. and C. Y. Suen [1984]. "A fast parallel algorithm for thinning digital patterns", *Commun. ACM*, vol. 7, no. 3, pp. 236-239.

- Zhang X. and Y. Xia [1983] "The automatic recognition of handprinted Chinese characters - a method of extracting an ordered sequence of strokes", *Pattern Recognition Letters*, vol. 1, pp. 259-265.
- Zhang Z., I. Hartmann, J. guo and R. Suchenwirth [1989]. "A recognition method of printed Chinese characters by features combination", *Int. Journal Of Research & Engineering, Postal Applications*, pp. 77-83.
- Zhao M. [1990]. "Two dimensional extended attributed grammar method for the recognition of handprinted Chinese characters", *Pattern Recognition*, vol. 23, pp. 685-695.



